Macro Basics for New SAS Users

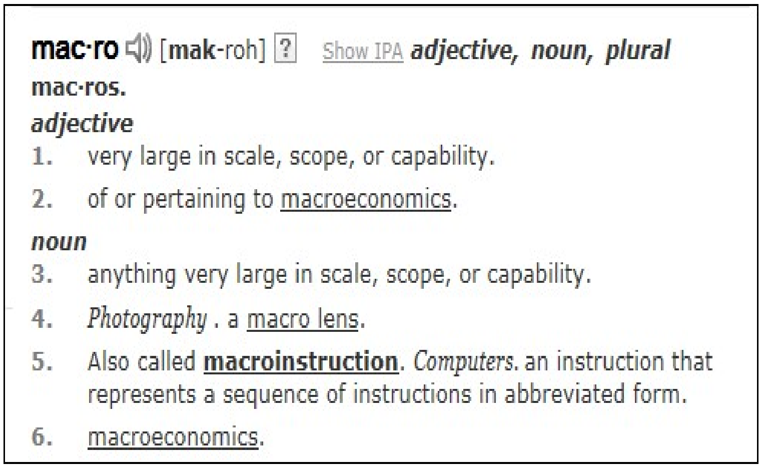
Michigan SAS User Group
By: Cynthia Zender

Today's Agenda

- Understand SAS Macro Facility
- Use macro variables for text substitution
- Use macro functions to alter text
- Use macro programs to generate code
- Show "bonus" material on interaction with DATA step and SQL

What Is "Macro" Anyway?

From: Dictionary.com



mac·ro [mak-roh] Show IPA *adjective, noun, plural* **mac·ros.**

adjective

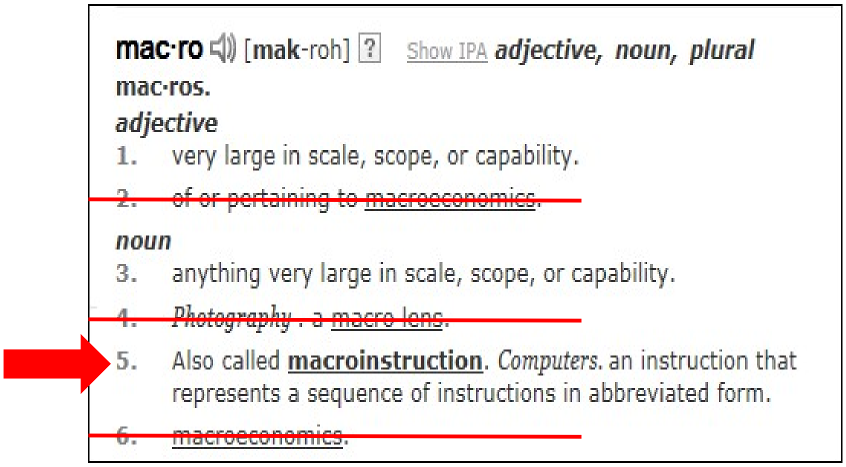
1. very large in scale, scope, or capability.
2. of or pertaining to macroeconomics.

noun

3. anything very large in scale, scope, or capability.
4. *Photography* . a macro lens.
5. Also called **macroinstruction**. *Computers.* an instruction that represents a sequence of instructions in abbreviated form.
6. macroeconomics.

What Is "Macro" Anyway?

From: Dictionary.com, #5 comes closest to the essence of the SAS Macro Facility



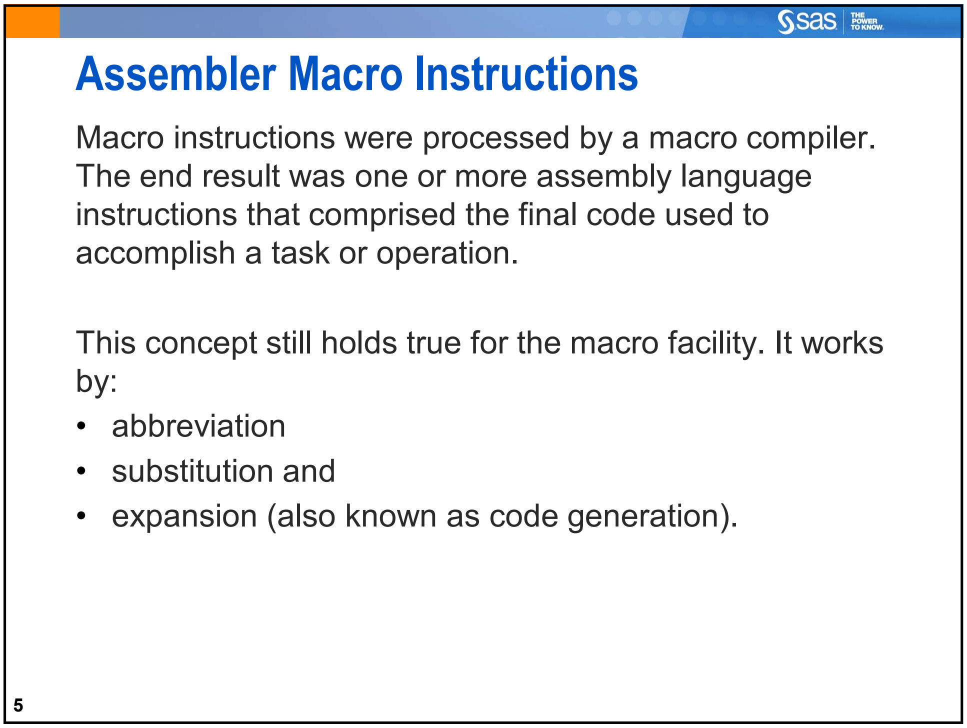
mac·ro [mak-roh] Show IPA *adjective, noun, plural* **mac·ros.**

adjective

1. very large in scale, scope, or capability.
2. ~~of or pertaining to macroeconomics.~~

noun

3. anything very large in scale, scope, or capability.
4. ~~*Photography* . a macro lens.~~
5. Also called **macroinstruction**. *Computers.* an instruction that represents a sequence of instructions in abbreviated form.
6. ~~macroeconomics.~~

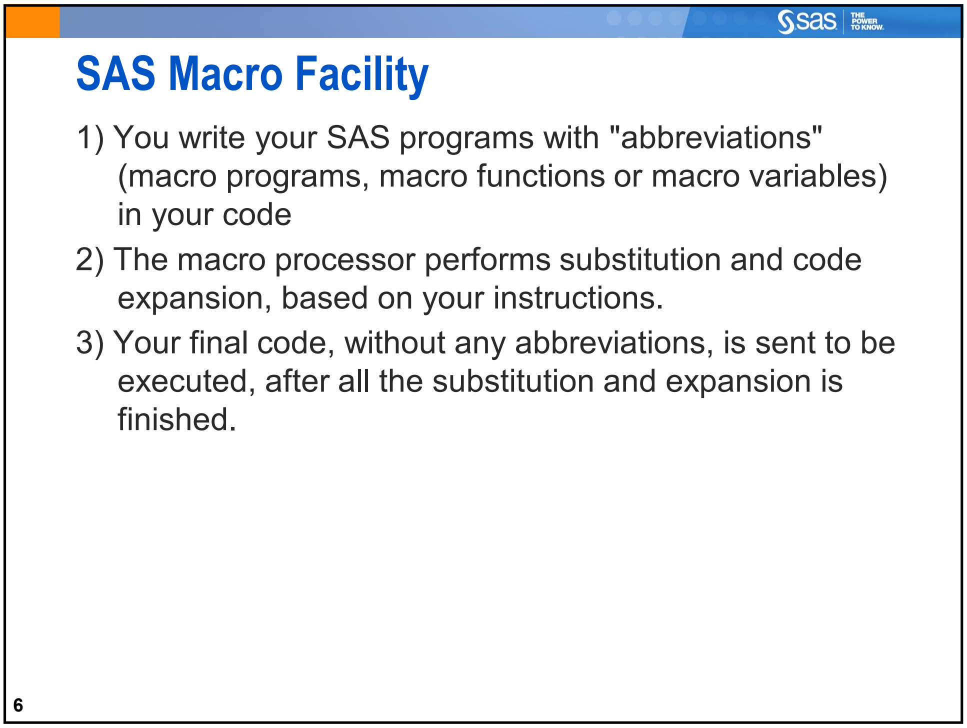


Assembler Macro Instructions

Macro instructions were processed by a macro compiler. The end result was one or more assembly language instructions that comprised the final code used to accomplish a task or operation.

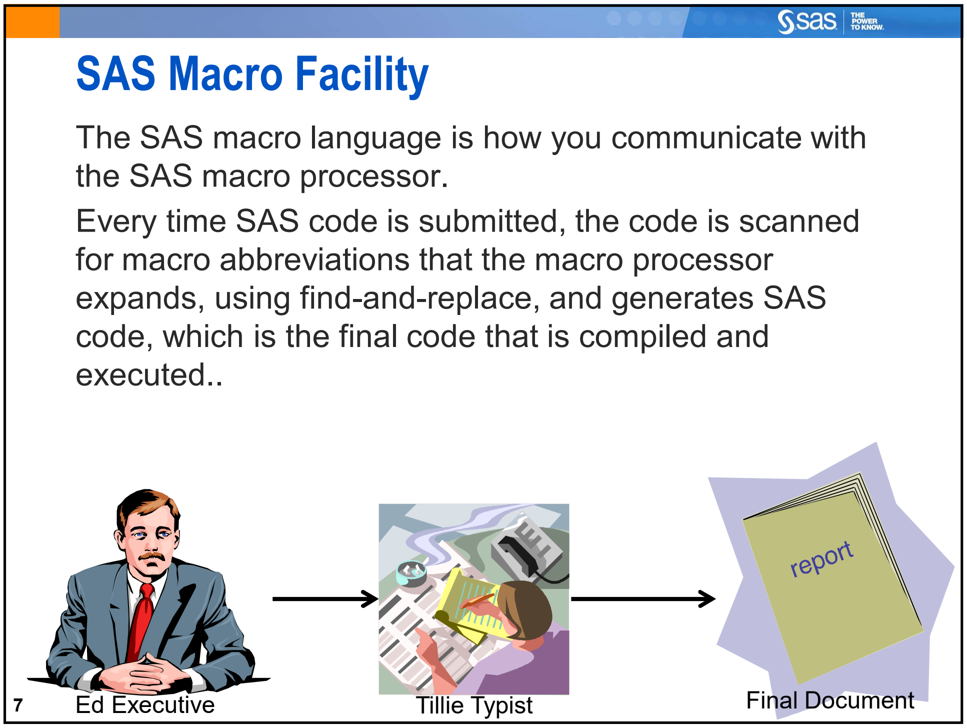
This concept still holds true for the macro facility. It works by:

- abbreviation
- substitution and
- expansion (also known as code generation).



SAS Macro Facility

1) You write your SAS programs with "abbreviations" (macro programs, macro functions or macro variables) in your code
2) The macro processor performs substitution and code expansion, based on your instructions.
3) Your final code, without any abbreviations, is sent to be executed, after all the substitution and expansion is finished.



SAS Macro Facility

The SAS macro language is how you communicate with the SAS macro processor.

Every time SAS code is submitted, the code is scanned for macro abbreviations that the macro processor expands, using find-and-replace, and generates SAS code, which is the final code that is compiled and executed..

Ed Executive

Tillie Typist

Final Document

Major Components of Macro Facility

1) Macro language
2) Macro processor

Special abbreviations in your code trigger the macro processor to do something.

The special abbreviations are the **&** and **%** symbols, which are called macro *triggers*.

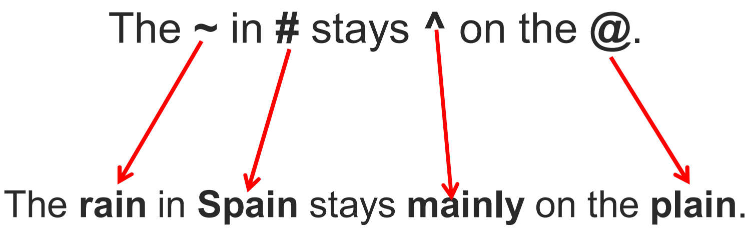
Macro Variables, Programs and Functions (Oh My!)

The **&** followed by a name (***&macvar***) is called a macro *variable*. The *&macvar* reference causes the macro processor do the equivalent of find-and-replace.

The **%** followed by a name (***%macname***) can call a predefined macro (either a macro program, a macro function, or a macro statement).

The macro functions or macro programs instruct the macro processor to perform a more elaborate find-and-replace with more code expansion and code generation than you can get with simple macro variable references.

Working with Find and Replace

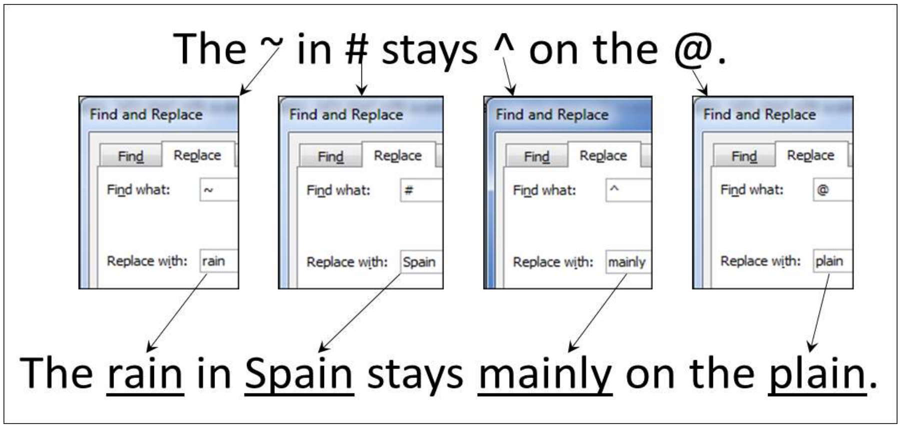


The **~** in **#** stays **^** on the **@**.

The **rain** in **Spain** stays **mainly** on the **plain**.

Working with Find and Replace

Using Find and Replace in Notepad:



The ~ in # stays ^ on the @.

Find what: ~
Replace with: rain

Find what: #
Replace with: Spain

Find what: ^
Replace with: mainly

Find what: @
Replace with: plain

The rain in Spain stays mainly on the plain.

Flexible Text Substitution

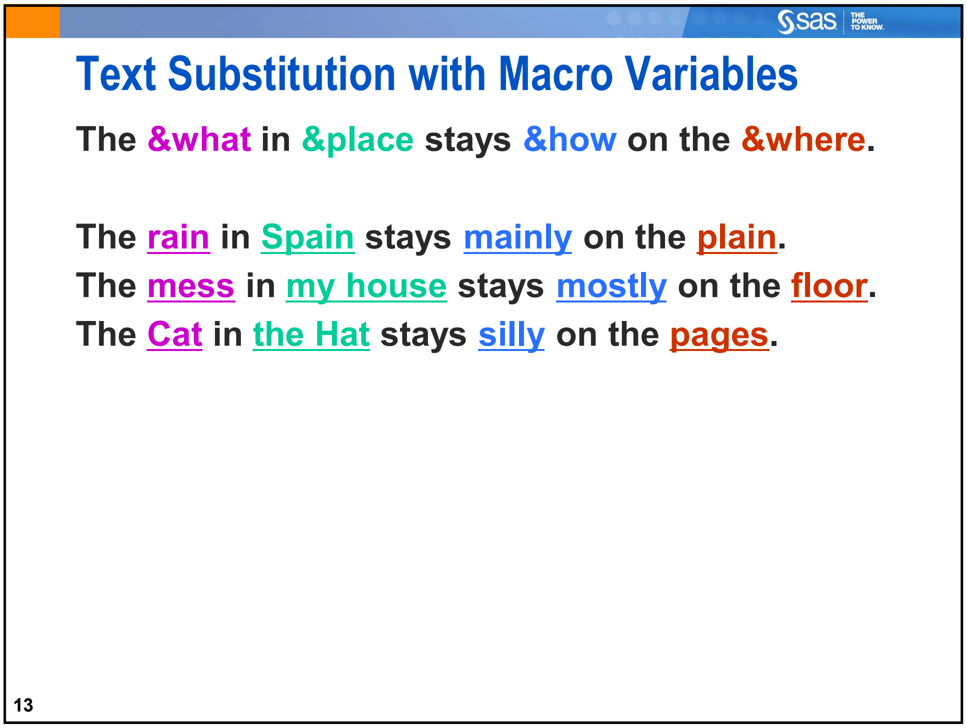
The ~ in # stays ^ on the @.

The rain in Spain stays mainly on the plain.

The mess in my house stays mostly on the floor.

The Cat in the Hat stays silly on the pages.

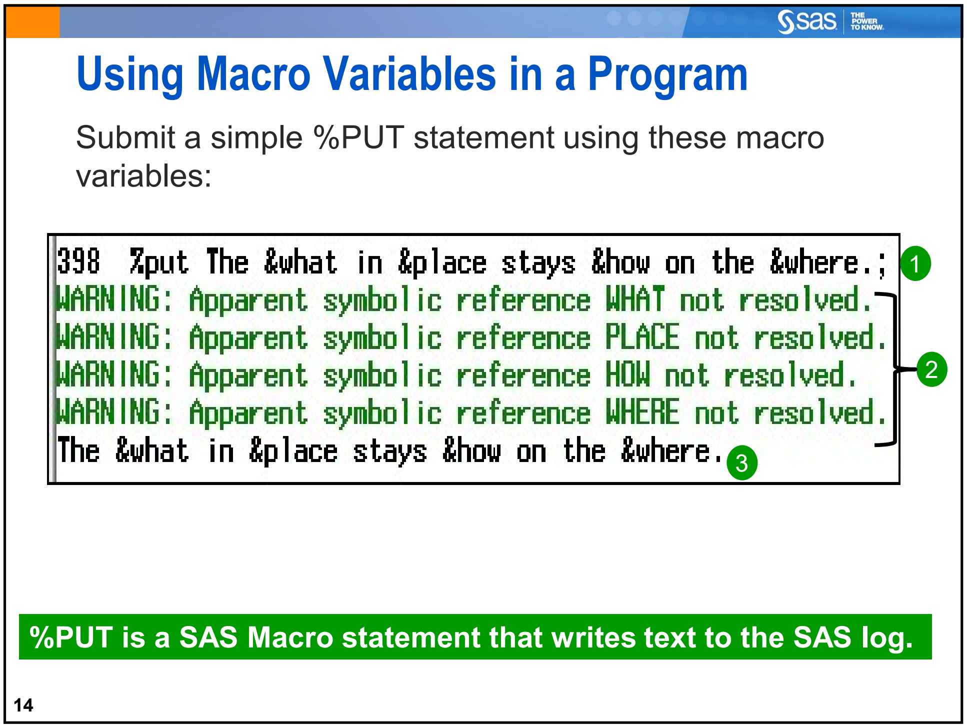
to the rescue!



Text Substitution with Macro Variables

The &what in &place stays &how on the &where.

The rain in Spain stays mainly on the plain.
The mess in my house stays mostly on the floor.
The Cat in the Hat stays silly on the pages.



Using Macro Variables in a Program

Submit a simple %PUT statement using these macro variables:

```
398  %put The &what in &place stays &how on the &where.;
WARNING: Apparent symbolic reference WHAT not resolved.
WARNING: Apparent symbolic reference PLACE not resolved.
WARNING: Apparent symbolic reference HOW not resolved.
WARNING: Apparent symbolic reference WHERE not resolved.
The &what in &place stays &how on the &where.
```

%PUT is a SAS Macro statement that writes text to the SAS log.

Behind the Scenes

1) The SAS macro processor scanned the %PUT statement text and tried to do substitution,.

2) The macro processor looked for values for the macro variables, but didn't find them. * So, the macro processor didn't know the proper values to substitute for &WHAT, &PLACE, &HOW, and &WHERE. The WARNING message is issued.

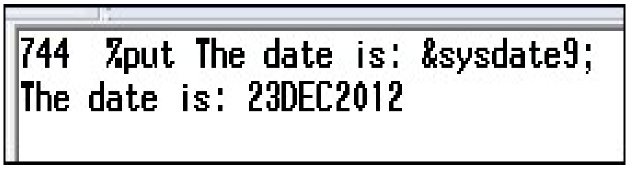
3) The %PUT statement wrote the unresolved text string into the SAS log.

* If the macro processor can find the values for the macro variables the results would be different.

Using a Different Macro Variable

With a different macro variable:

```
%put The date is: &sysdate9;
```



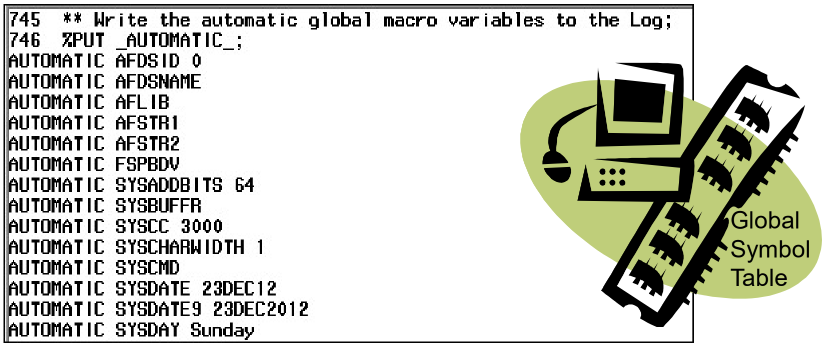
```
744  %put The date is: &sysdate9;
The date is: 23DEC2012
```

&SYSDATE9 is an "automatic" macro variable created by SAS when a SAS session starts.

Where Are Macro Variables Stored?

SAS macro variables "live" in a memory table called the Global Symbol Table. To see the list of all the global macro variables:

```
%PUT _AUTOMATIC_;
```



```
745  ** Write the automatic global macro variables to the Log;
746  %PUT _AUTOMATIC_;
AUTOMATIC AFDSID 0
AUTOMATIC AFDSNAME
AUTOMATIC AFLIB
AUTOMATIC AFSTR1
AUTOMATIC AFSTR2
AUTOMATIC FSPBDV
AUTOMATIC SYSADDBITS 64
AUTOMATIC SYSBUFFR
AUTOMATIC SYSCC 3000
AUTOMATIC SYSCHARWIDTH 1
AUTOMATIC SYSCMD
AUTOMATIC SYSDATE 23DEC12
AUTOMATIC SYSDATE9 23DEC2012
AUTOMATIC SYSDAY Sunday
```

Creating Macro Variables

Use the %LET statement to create macro variables:

```
%LET macvar = value;
```

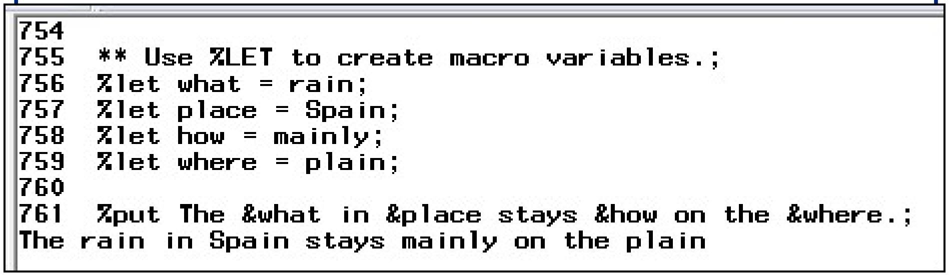
- ***macvar*** is the name of the macro variable that you want to create
- ***value*** is the value you want to have entered in the Global Symbol Table.

Revised Program

```
** Use %LET to create macro variables.;
%let what = rain;
%let place = Spain;
%let how = mainly;
%let where = plain;
%put The &what in &place stays &how on the &where.;
```

Revised Program

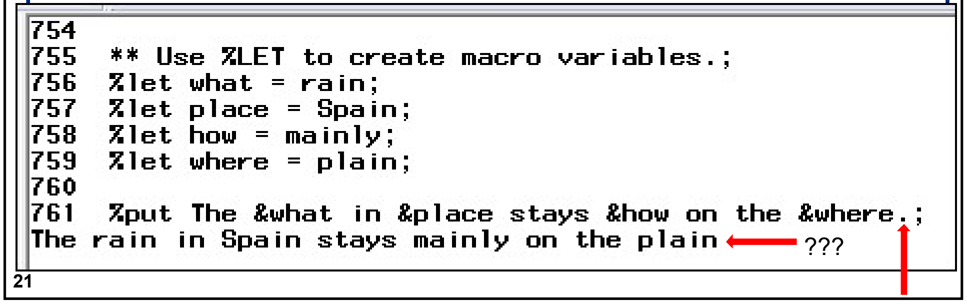
```
** Use %LET to create macro variables.;
%let what = rain;
%let place = Spain;
%let how = mainly;
%let where = plain;
%put The &what in &place stays &how on the &where.;
```



```
754
755   ** Use %LET to create macro variables.;
756   %let what = rain;
757   %let place = Spain;
758   %let how = mainly;
759   %let where = plain;
760
761   %put The &what in &place stays &how on the &where.;
The rain in Spain stays mainly on the plain
```

Revised Program

```
** Use %LET to create macro variables.;
%let what = rain;
%let place = Spain;
%let how = mainly;
%let where = plain;
%put The &what in &place stays &how on the &where.;
```

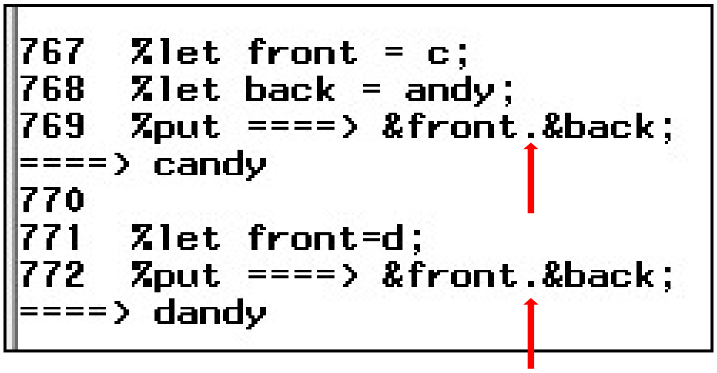


```
754
755   ** Use %LET to create macro variables.;
756   %let what = rain;
757   %let place = Spain;
758   %let how = mainly;
759   %let where = plain;
760
761   %put The &what in &place stays &how on the &where.;
The rain in Spain stays mainly on the plain
```

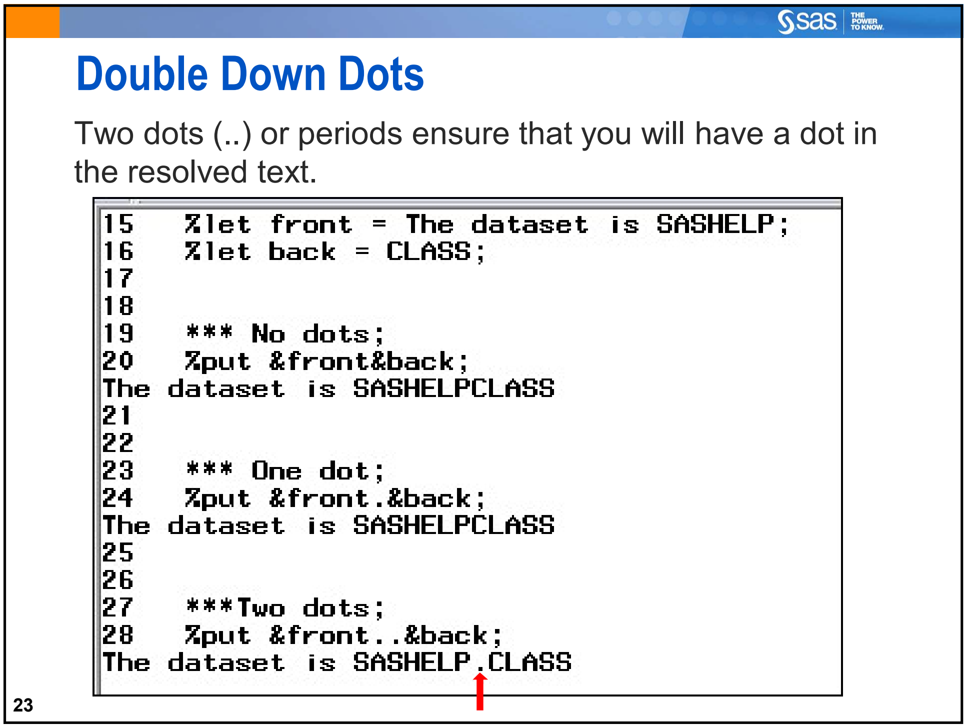
???

Dots are Delimiters

A dot (.) or period delimits the end of a macro variable, so that two macro variables can be concatenated together.



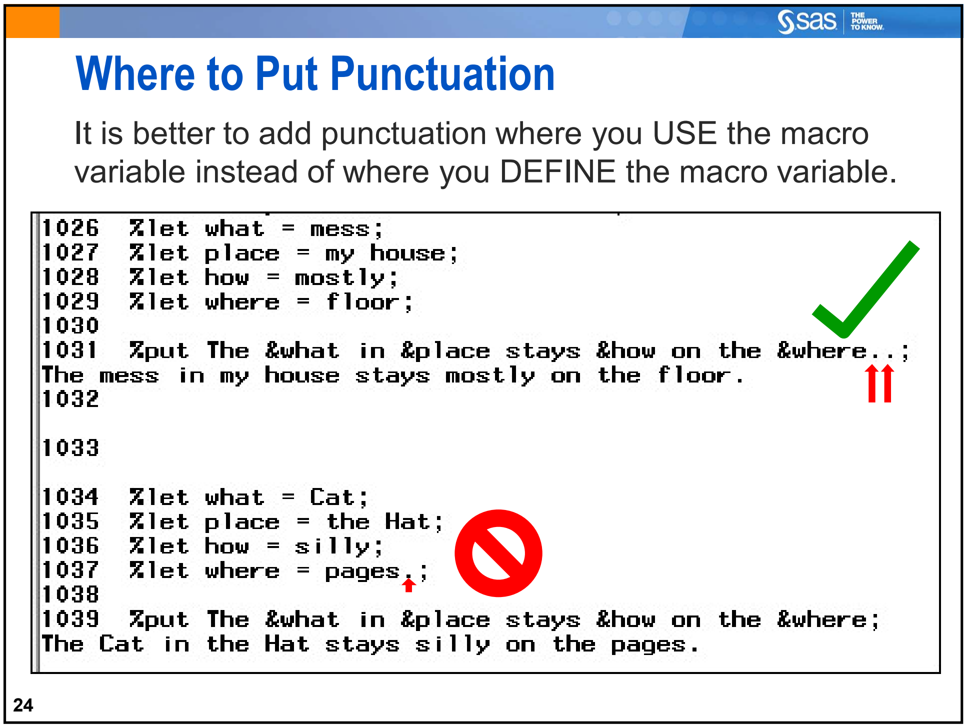
```
767   %let front = c;
768   %let back = andy;
769   %put ====> &front.&back;
====> candy
770
771   %let front=d;
772   %put ====> &front.&back;
====> dandy
```



Double Down Dots

Two dots (..) or periods ensure that you will have a dot in the resolved text.

```
15    %let front = The dataset is SASHELP;
16    %let back = CLASS;
17
18
19    *** No dots;
20    %put &front&back;
The dataset is SASHELPCLASS
21
22
23    *** One dot;
24    %put &front.&back;
The dataset is SASHELPCLASS
25
26
27    ***Two dots;
28    %put &front..&back;
The dataset is SASHELP.CLASS
```



Where to Put Punctuation

It is better to add punctuation where you USE the macro variable instead of where you DEFINE the macro variable.

```
1026  %let what = mess;
1027  %let place = my house;
1028  %let how = mostly;
1029  %let where = floor;
1030
1031  %put The &what in &place stays &how on the &where..;
The mess in my house stays mostly on the floor.
1032

1033

1034  %let what = Cat;
1035  %let place = the Hat;
1036  %let how = silly;
1037  %let where = pages.;
1038
1039  %put The &what in &place stays &how on the &where;
The Cat in the Hat stays silly on the pages.
```

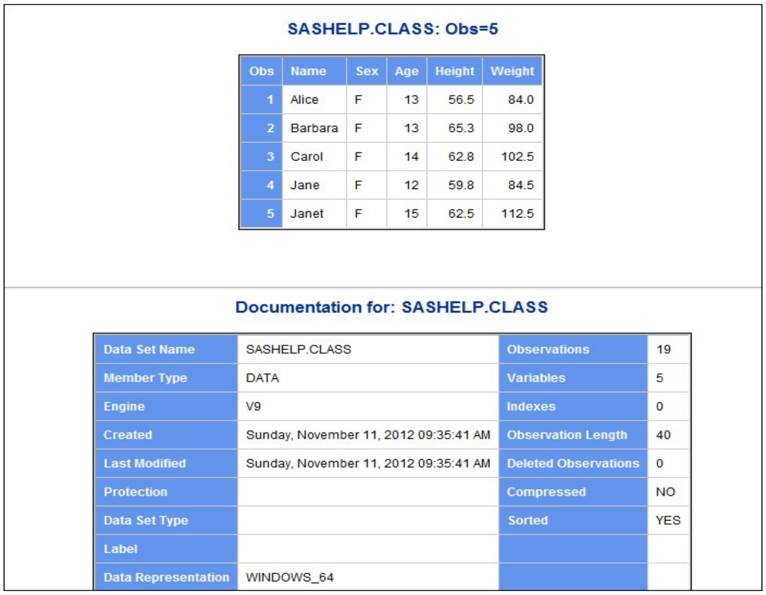
Yawn!

Show me more!

Starting Program

```
ods html file='c:\temp\report.html'
         style=sasweb;
  proc print data=sashelp.class(obs=5);
    title "SASHELP.CLASS: Obs=5";
  run;

  ods select attributes variables sortedby;
  proc contents data=sashelp.class;
    title "Documentation for: SASHELP.CLASS";
  run;
ods html close;
```

Partial Results



SASHELP.CLASS: Obs=5

Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0
2	Barbara	F	13	65.3	98.0
3	Carol	F	14	62.8	102.5
4	Jane	F	12	59.8	84.5
5	Janet	F	15	62.5	112.5

Documentation for: SASHELP.CLASS

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Sunday, November 11, 2012 09:35:41 AM	Observation Length	40
Last Modified	Sunday, November 11, 2012 09:35:41 AM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label			
Data Representation	WINDOWS_64		

Macro-ize the Code

To "macro-ize" the code means to select code snippets that are good candidates for "find and replace" by the macro processor.

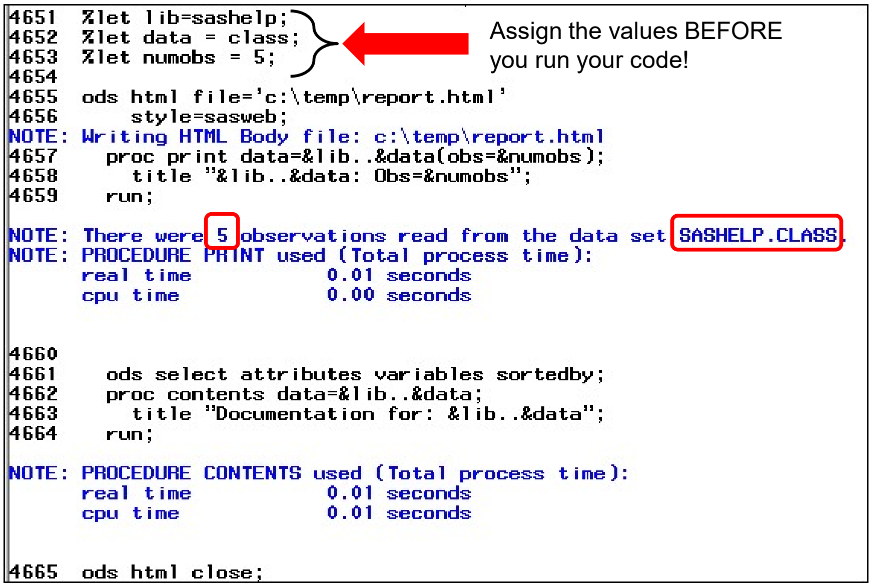
```
ods html file='c:\temp\report.html' style=sasweb;
  proc print data=sashelp.class(obs=5);
    title "SASHELP.CLASS: Obs=5";
  run;

  ods select attributes variables sortedby;
  proc contents data=sashelp.class;
    title "Documentation for: SASHELP.CLASS";
  run;
ods html close;
```

Insert Macro Variables in the Code

Of course, you have to use the same macro variable names throughout the code for substitution and resolution to work correctly.

```
ods html file='c:\temp\report.html' style=sasweb;
  proc print data=&lib..&data(obs=&numobs);
    title "&lib..&data: Obs=&numobs";
  run;

  ods select attributes variables sortedby;
  proc contents data=&lib..&data;
    title "Documentation for: &lib..&data";
  run;
ods html close;
```

Use %LET To Assign Values

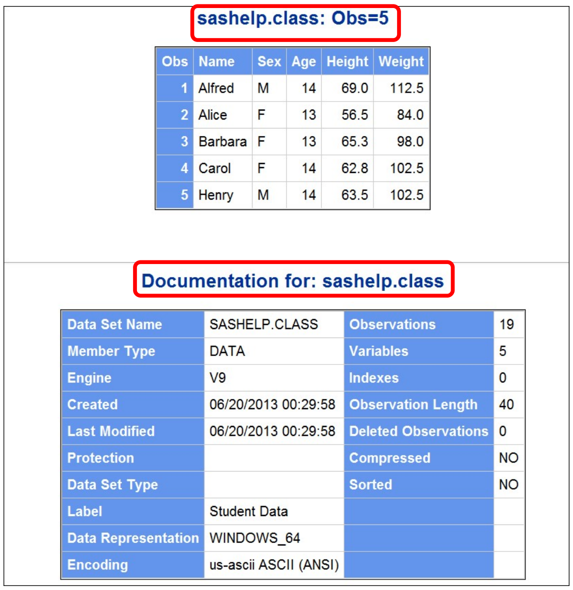


```
4651   %let lib=sashelp;
4652   %let data = class;
4653   %let numobs = 5;
4654
4655   ods html file='c:\temp\report.html'
4656       style=sasweb;
NOTE: Writing HTML Body file: c:\temp\report.html
4657     proc print data=&lib..&data(obs=&numobs);
4658       title "&lib..&data: Obs=&numobs";
4659     run;

NOTE: There were 5 observations read from the data set SASHELP.CLASS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds


4660
4661     ods select attributes variables sortedby;
4662     proc contents data=&lib..&data;
4663       title "Documentation for: &lib..&data";
4664     run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


4665   ods html close;
```

Assign the values BEFORE you run your code!

Partial Results

Note the titles are in lowercase:



sashelp.class: Obs=5

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5

Documentation for: sashelp.class

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	06/20/2013 00:29:58	Observation Length	40
Last Modified	06/20/2013 00:29:58	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_64		
Encoding	us-ascii ASCII (ANSI)		

How the Text Was Resolved

Notice that the string 'sashelp.class' was resolved and inserted just as it was typed (in lower case).

What Was Sent to the Macro Processor

```
ods html file='c:\temp\report.html'
    style=sasweb;
  proc print data=&lib..&data
             (obs=&numobs);
   title "&lib..&data: Obs=&numobs";
  run;

ods select attributes variables sortedby;
 proc contents data=&lib..&data;
  title "Documentation for: &lib..&data";
 run;
ods html close;
```

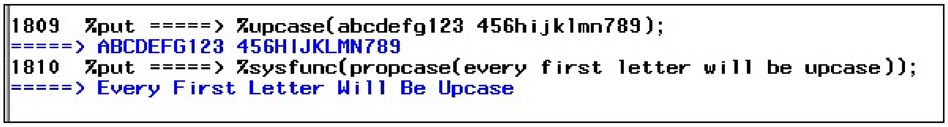
The Resolved Text

```
ods html file='c:\temp\report.html'
    style=sasweb;
  proc print data=sashelp.class
             (obs=5);
   title "sashelp.class: Obs=5";
  run;

ods select attributes variables sortedby;
 proc contents data=sashelp.class;
  title "Documentation for: sashelp.class";
 run;
ods html close;
```

The macro processor will not alter the text that is assigned with a %LET statement. If the values need to be changed inside the macro processor before the replaced text is sent to the compiler, use a macro function or %SYSFUNC.

Changing Macro Variable Values

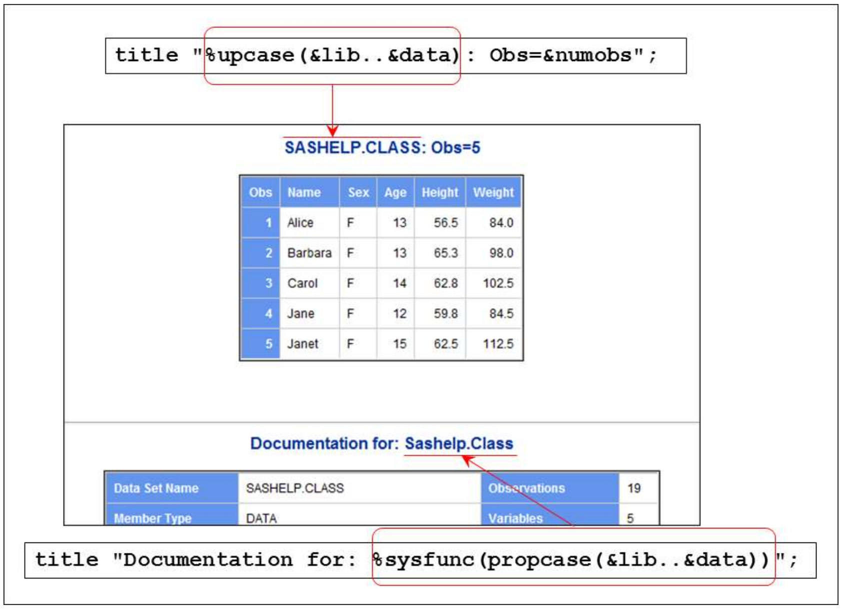
Use macro functions to change macro variable values.



```
1809  %put =====> %upcase(abcdefg123 456hijklmn789);
=====> ABCDEFG123 456HIJKLMN789
1810  %put =====> %sysfunc(propcase(every first letter will be upcase));
=====> Every First Letter Will Be Upcase
```

If there is not a macro function to do what you need, you can use %SYSFUNC to call a DATA step function (like PROPCASE).

Changed Results



```
title "%upcase(&lib..&data): Obs=&numobs";
```

SASHELP.CLASS: Obs=5

Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0
2	Barbara	F	13	65.3	98.0
3	Carol	F	14	62.8	102.5
4	Jane	F	12	59.8	84.5
5	Janet	F	15	62.5	112.5

Documentation for: Sashelp.Class

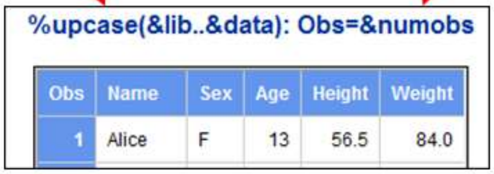
Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5

```
title "Documentation for: %sysfunc(propcase(&lib..&data))";
```

Danger, Will Robinson, Danger!

Macro variables will not resolve inside single quotes.

```
title '%upcase(&lib..&data): Obs=&numobs';
```



%upcase(&lib..&data): Obs=&numobs

Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0

Any Other "Doubles" to Worry About?

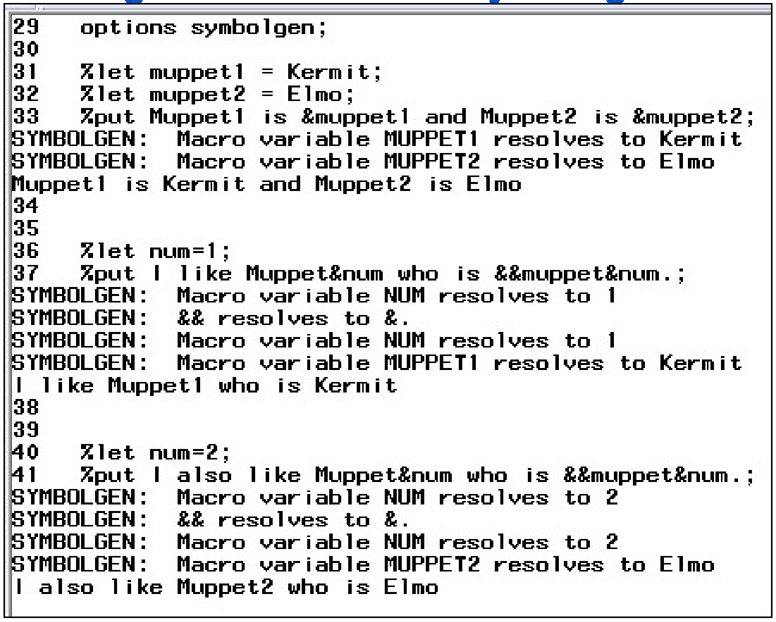
Sometimes, you may want to use multiple ampersands for the indirect reference to macro variables.

```
138   **5) Example of indirect reference;
139   %let muppet1 = Kermit;
140   %let muppet2 = Elmo;
141   %put Muppet1 is &muppet1 and Muppet2 is &muppet2;
Muppet1 is Kermit and Muppet2 is Elmo
142
143   %let num=1;
144   %put I like Muppet&num who is &&muppet&num;
I like Muppet1 who is Kermit
145
146   %let num=2;
147   %put I also like Muppet&num who is &&muppet&num;
I also like Muppet2 who is Elmo
```

What?!?

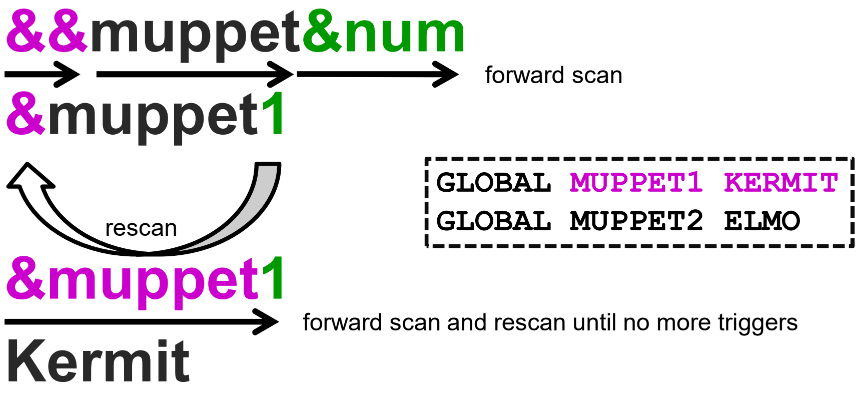
&&muppet&num?

Tracking Resolution with Symbolgen



```
29   options symbolgen;
30
31   %let muppet1 = Kermit;
32   %let muppet2 = Elmo;
33   %put Muppet1 is &muppet1 and Muppet2 is &muppet2;
SYMBOLGEN:  Macro variable MUPPET1 resolves to Kermit
SYMBOLGEN:  Macro variable MUPPET2 resolves to Elmo
Muppet1 is Kermit and Muppet2 is Elmo
34
35
36   %let num=1;
37   %put I like Muppet&num who is &&muppet&num.;
SYMBOLGEN:  Macro variable NUM resolves to 1
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable NUM resolves to 1
SYMBOLGEN:  Macro variable MUPPET1 resolves to Kermit
I like Muppet1 who is Kermit
38
39
40   %let num=2;
41   %put I also like Muppet&num who is &&muppet&num.;
SYMBOLGEN:  Macro variable NUM resolves to 2
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable NUM resolves to 2
SYMBOLGEN:  Macro variable MUPPET2 resolves to Elmo
I also like Muppet2 who is Elmo
```

Forward Scan Rule and Rescan Rule



&&muppet&num

forward scan

&muppet1

rescan

```
GLOBAL MUPPET1 KERMIT
GLOBAL MUPPET2 ELMO
```

&muppet1

forward scan and rescan until no more triggers

Kermit

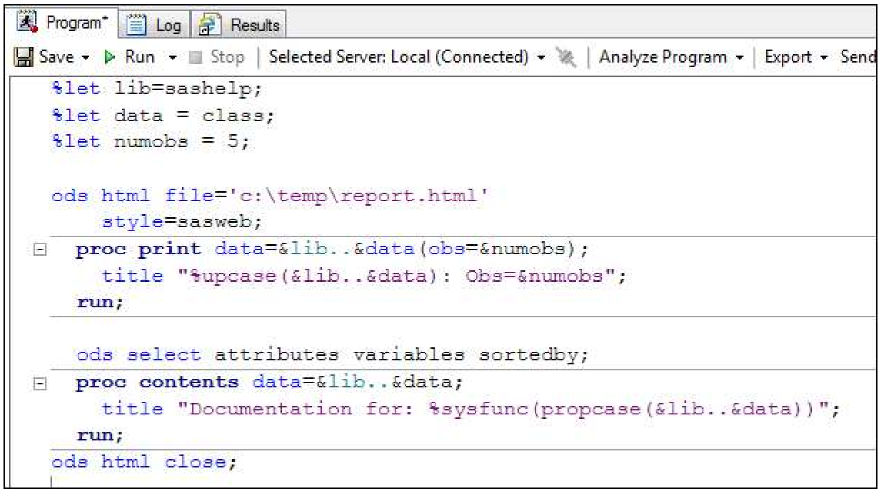
%put I like Muppet&num who is &&muppet&num.;

becomes

I like Muppet1 who is Kermit

Do Macro Variables Work in Enterprise Guide?

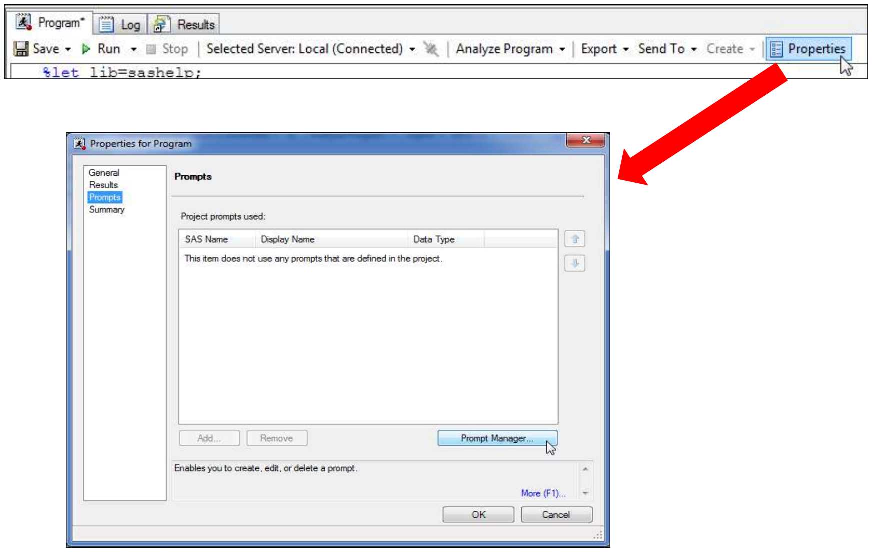
Absolutely!



```
%let lib=sashelp;
%let data = class;
%let numobs = 5;

ods html file='c:\temp\report.html'
    style=sasweb;
  proc print data=&lib..&data(obs=&numobs);
    title "%upcase(&lib..&data): Obs=&numobs";
  run;

  ods select attributes variables sortedby;
  proc contents data=&lib..&data;
    title "Documentation for: %sysfunc(propcase(&lib..&data))";
  run;
ods html close;
```

Properties and Prompt Manager

Click Properties to define prompts (a.k.a macro variables)



```
%let lib=sashelp;
```

Properties for Program

Prompts

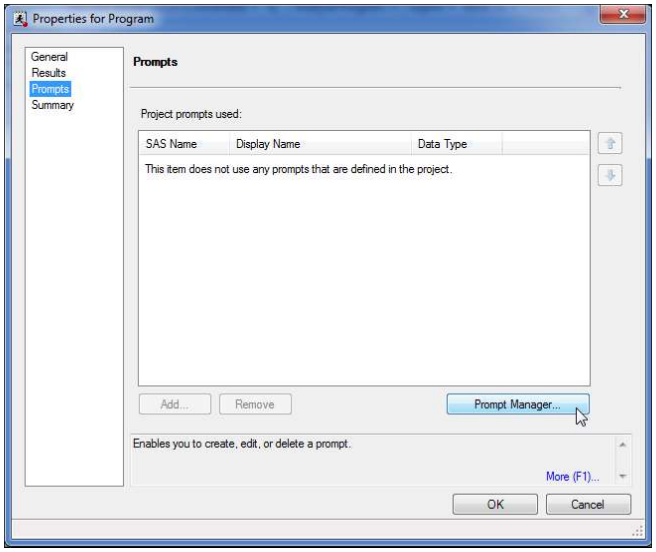
Project prompts used:

SAS Name	Display Name	Data Type
This item does not use any prompts that are defined in the project.		

Enables you to create, edit, or delete a prompt.

Properties and Prompt Manager

Click Properties to define prompts (a.k.a macro variables)



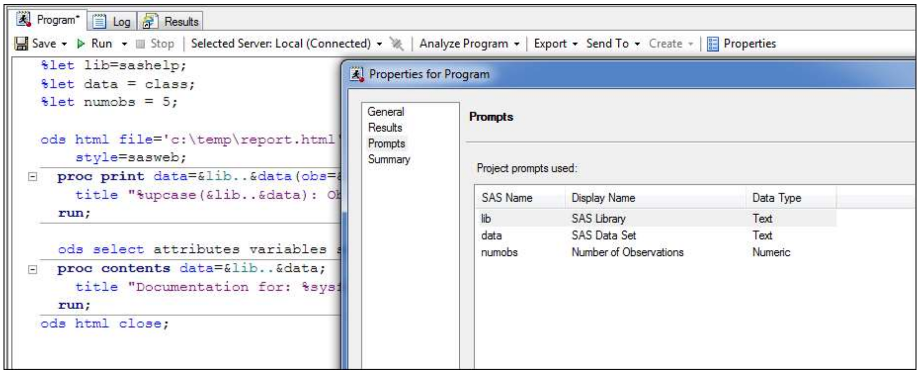
Prompts

Project prompts used:

SAS Name	Display Name	Data Type
This item does not use any prompts that are defined in the project.		

Enables you to create, edit, or delete a prompt.

Add the Macro Variables as Prompts



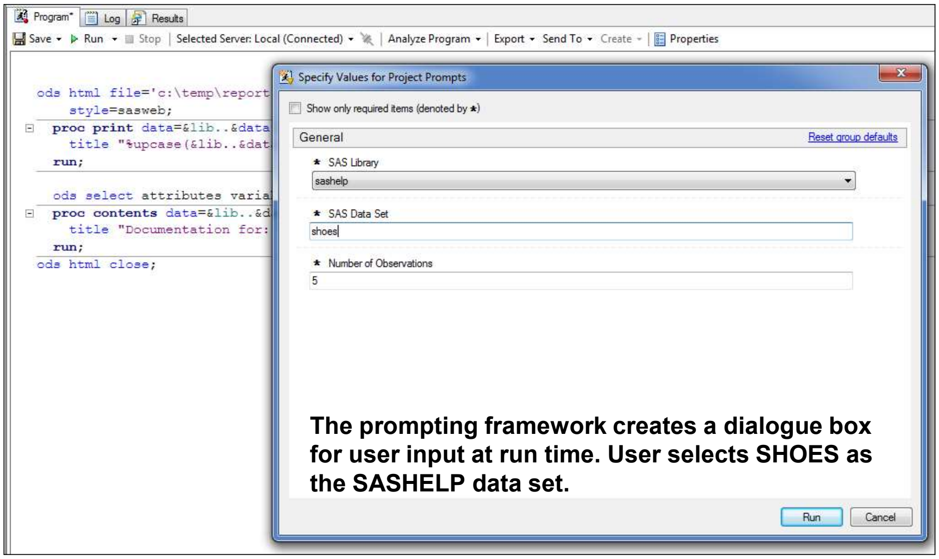
```
%let lib=sashelp;
%let data = class;
%let numobs = 5;

ods html file='c:\temp\report.html
    style=sasweb;
 proc print data=&lib..&data(obs=
   title "%upcase(&lib..&data): O
 run;

 ods select attributes variables
 proc contents data=&lib..&data;
   title "Documentation for: %sys
 run;
ods html close;
```

Prompts

Project prompts used:

SAS Name	Display Name	Data Type
lib	SAS Library	Text
data	SAS Data Set	Text
numobs	Number of Observations	Numeric

Run the Program



```
ods html file='c:\temp\report
    style=sasweb;
  proc print data=&lib..&data
    title "%upcase(&lib..&dat
  run;

  ods select attributes varia
  proc contents data=&lib..&d
    title "Documentation for:
  run;
ods html close;
```

☐ Show only required items (denoted by ★)

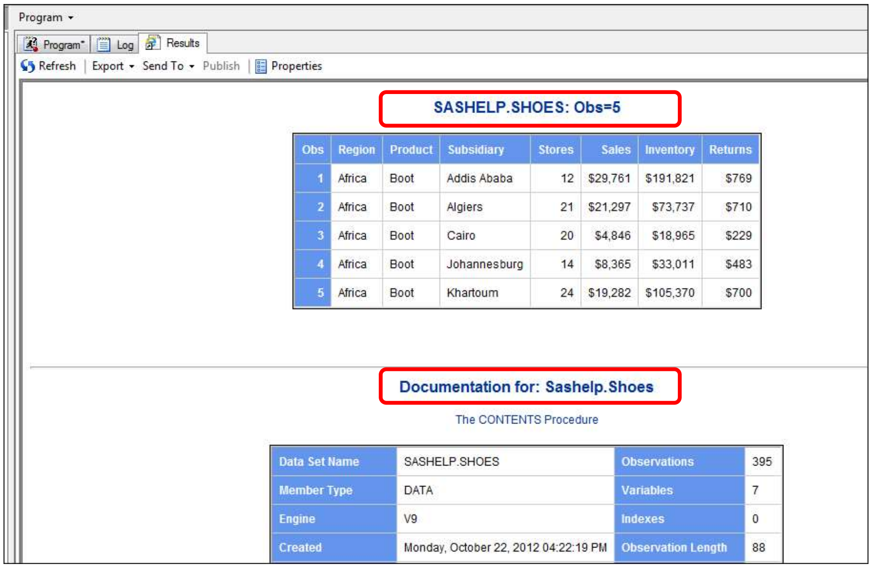
★ SAS Library: sashelp

★ SAS Data Set: shoes

★ Number of Observations: 5

The prompting framework creates a dialogue box for user input at run time. User selects SHOES as the SASHELP data set.

Partial Results



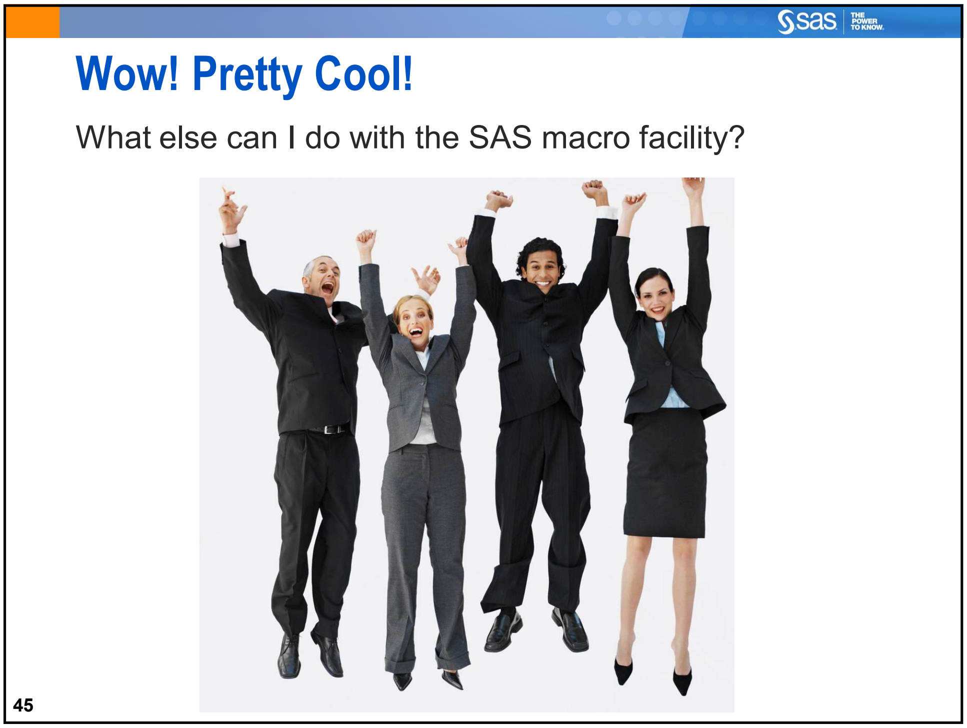
SASHELP.SHOES: Obs=5

Obs	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	$29,761	$191,821	$769
2	Africa	Boot	Algiers	21	$21,297	$73,737	$710
3	Africa	Boot	Cairo	20	$4,846	$18,965	$229
4	Africa	Boot	Johannesburg	14	$8,365	$33,011	$483
5	Africa	Boot	Khartoum	24	$19,282	$105,370	$700

Documentation for: Sashelp.Shoes

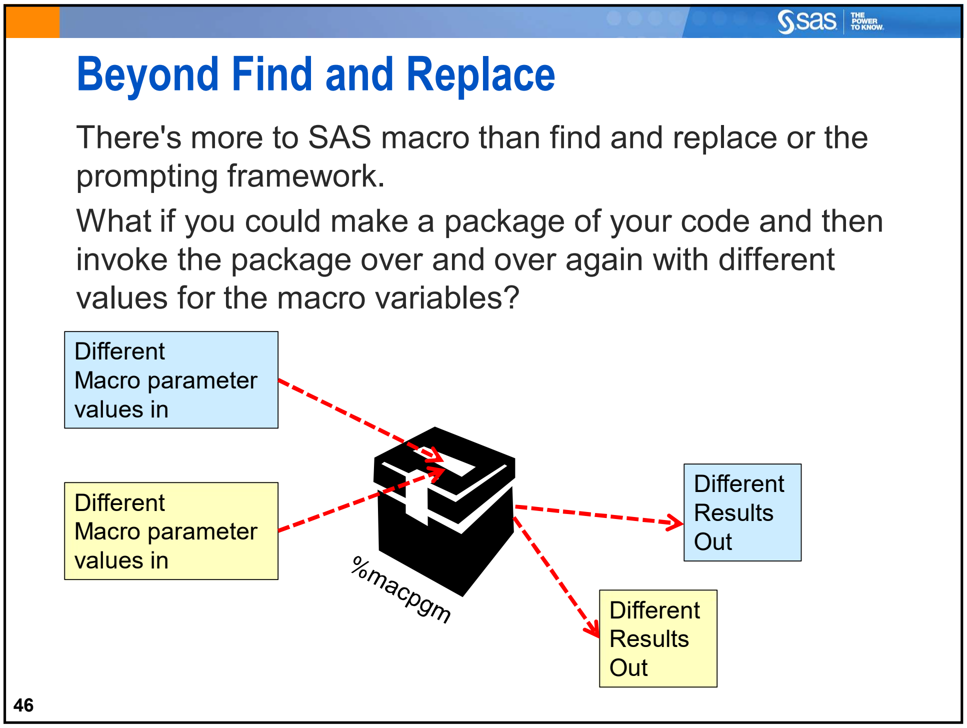
The CONTENTS Procedure

Data Set Name	SASHELP.SHOES	Observations	395
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	Monday, October 22, 2012 04:22:19 PM	Observation Length	88



Wow! Pretty Cool!

What else can I do with the SAS macro facility?



Beyond Find and Replace

There's more to SAS macro than find and replace or the prompting framework.

What if you could make a package of your code and then invoke the package over and over again with different values for the macro variables?

Different Macro parameter values in

Different Macro parameter values in

%macpgm

Different Results Out

Different Results Out

Some Possibilities

What if you want to allow users to select a different library and data set for the program that's been macro-ized?

And you also want them to have the ability to choose to get the PROC PRINT output by itself, the PROC CONTENTS output by itself, or both outputs?

And if they ask for a particular data set (SASHELP.CLASS), you want to assign a value of 19 to &NUMOBS.

Or, you are not ready to let them select the number of observations yet, so you want to put the parameter in the macro program, but for the short term, override whatever they select to 5 as the number.

IF and DO

The IF and DO statements belong in the SAS programming language interface (DATA step program).

To perform conditional processing that performs different text substitution, you need to use %IF and %DO.

```
%if &lib = sashelp and &data = class %then
    %let numobs = 19;
%else %let numobs = 5;
```

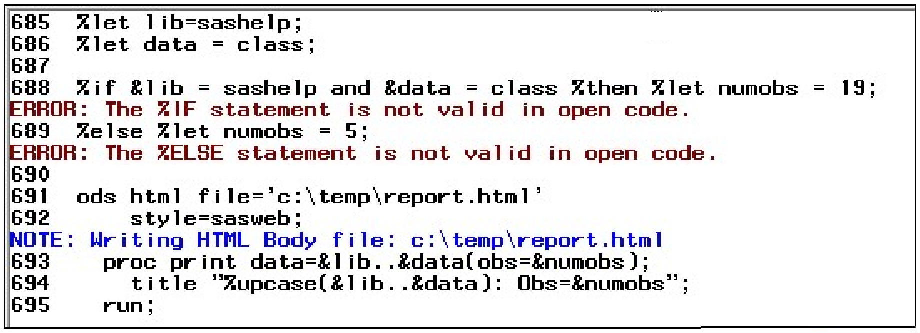
IF and DO vs %IF and %DO

The IF and DO statements belong in the SAS programming language interface (DATA step program).

To use conditional processing with macro text substitution, you need to use %IF and %DO.

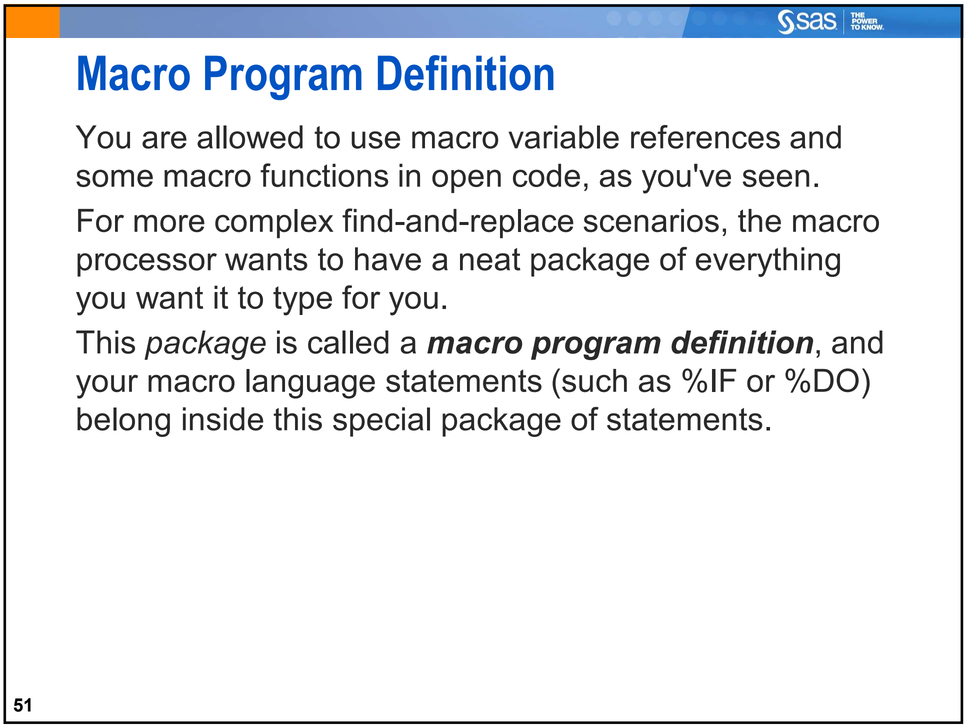
```
%if &lib = sashelp and &data = class %then
    %let numobs = 19;
%else %let numobs = 5;

ods html file='c:\temp\report.html'
    style=sasweb;
. . . same code . . .
ods html close;
```

Not Valid in Open Code



```
685  %let lib=sashelp;
686  %let data = class;
687
688  %if &lib = sashelp and &data = class %then %let numobs = 19;
ERROR: The %IF statement is not valid in open code.
689  %else %let numobs = 5;
ERROR: The %ELSE statement is not valid in open code.
690
691  ods html file='c:\temp\report.html'
692      style=sasweb;
NOTE: Writing HTML Body file: c:\temp\report.html
693    proc print data=&lib..&data(obs=&numobs);
694      title "%upcase(&lib..&data): Obs=&numobs";
695    run;
```

What happens in macro stays in macro!!!
(until 9.4M5 then you can use simple %IF/%DO/%END in open code)

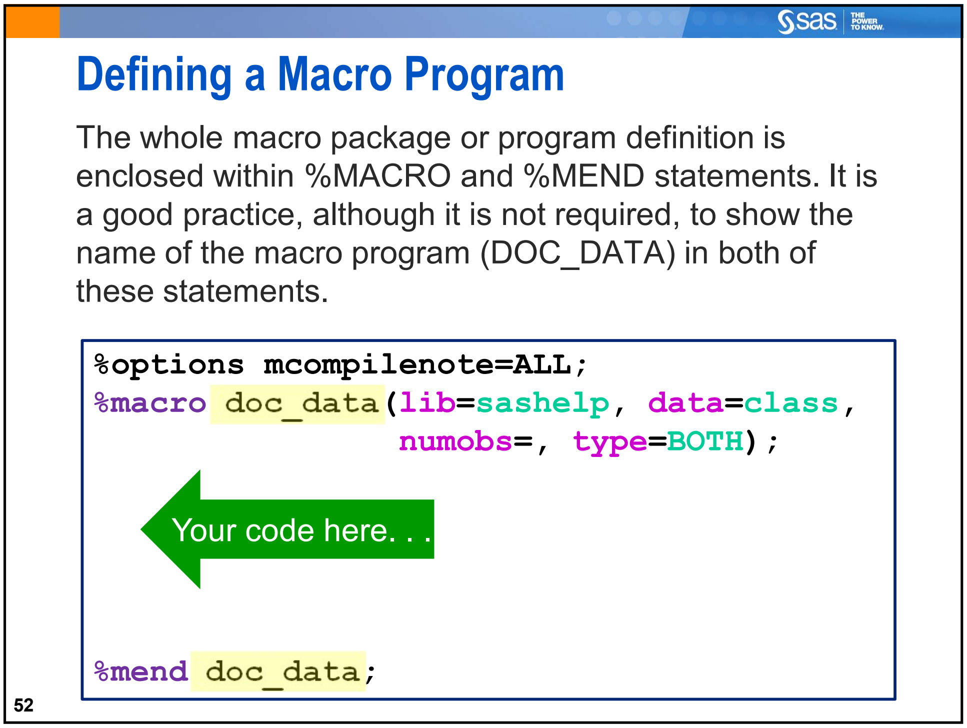


Macro Program Definition

You are allowed to use macro variable references and some macro functions in open code, as you've seen.

For more complex find-and-replace scenarios, the macro processor wants to have a neat package of everything you want it to type for you.

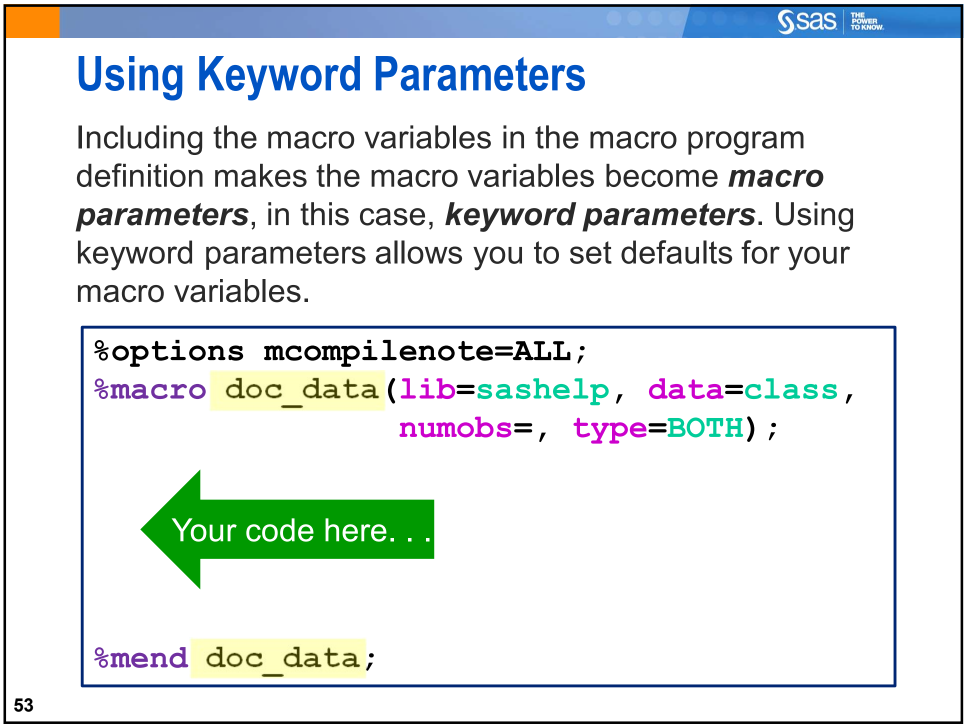
This *package* is called a ***macro program definition***, and your macro language statements (such as %IF or %DO) belong inside this special package of statements.



Defining a Macro Program

The whole macro package or program definition is enclosed within %MACRO and %MEND statements. It is a good practice, although it is not required, to show the name of the macro program (DOC_DATA) in both of these statements.

```
%options mcompilenote=ALL;
%macro doc_data(lib=sashelp, data=class,
                numobs=, type=BOTH);

     Your code here. . .

%mend doc_data;
```



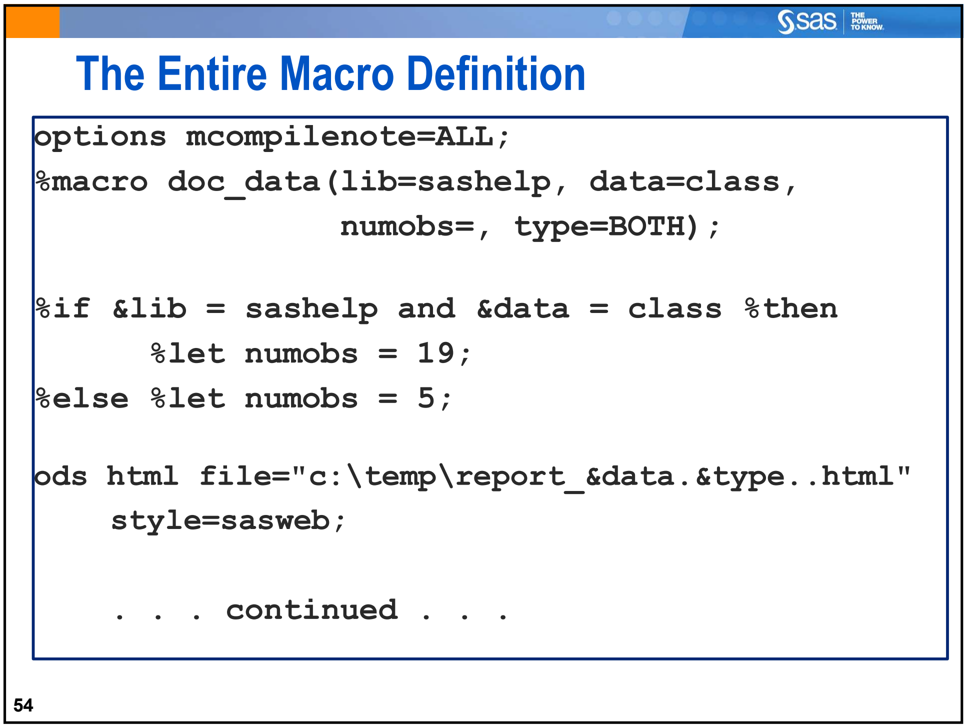
Using Keyword Parameters

Including the macro variables in the macro program definition makes the macro variables become ***macro parameters***, in this case, ***keyword parameters***. Using keyword parameters allows you to set defaults for your macro variables.

```
%options mcompilenote=ALL;
%macro doc_data(lib=sashelp, data=class,
                numobs=, type=BOTH);
```

Your code here. . .

```
%mend doc_data;
```



The Entire Macro Definition

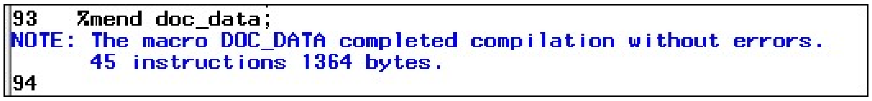
```
options mcompilenote=ALL;
%macro doc_data(lib=sashelp, data=class,
                numobs=, type=BOTH);

%if &lib = sashelp and &data = class %then
      %let numobs = 19;
%else %let numobs = 5;

ods html file="c:\temp\report_&data.&type..html"
    style=sasweb;

    . . . continued . . .
```

The Entire Macro Definition

```
%if &type = P or &type = BOTH %then %do;
  proc print data=&lib..&data(obs=&numobs);
  run;
%end;
%if &type = C or &type=BOTH %then %do;
  ods select attributes variables sortedby;
  proc contents data=&lib..&data;
  run;
%end;
ods html close;
%mend doc_data;
```

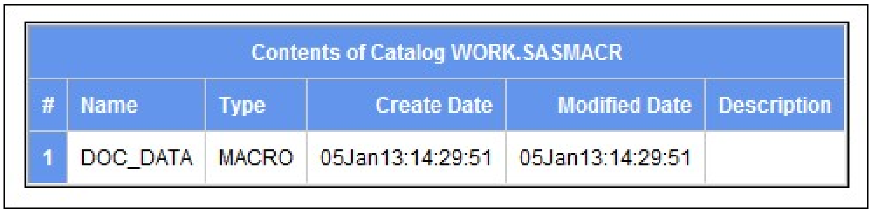
Successful Macro Compile Phase

The MCOMPILENOTE=ALL option puts a note in the log about macro compilation.



```
93   %mend doc_data;
NOTE: The macro DOC_DATA completed compilation without errors.
      45 instructions 1364 bytes.
94
```

The session-compiled macro program is stored in a SAS catalog called WORK.SASMACR



Contents of Catalog WORK.SASMACR					
#	Name	Type	Create Date	Modified Date	Description
1	DOC_DATA	MACRO	05Jan13:14:29:51	05Jan13:14:29:51	

Calling or Invoking the Macro Program

How do you make the macro processor do something with the package or macro definition? To call or invoke the macro program, use the percent sign again, but with the macro program name:

```
%DOC_DATA(. . .specify macro variable values . . .)
```

When this call to the %DOC_DATA macro program is submitted, the % is a macro trigger that causes the macro processor to spring into action, and . . .

Behind the Scenes

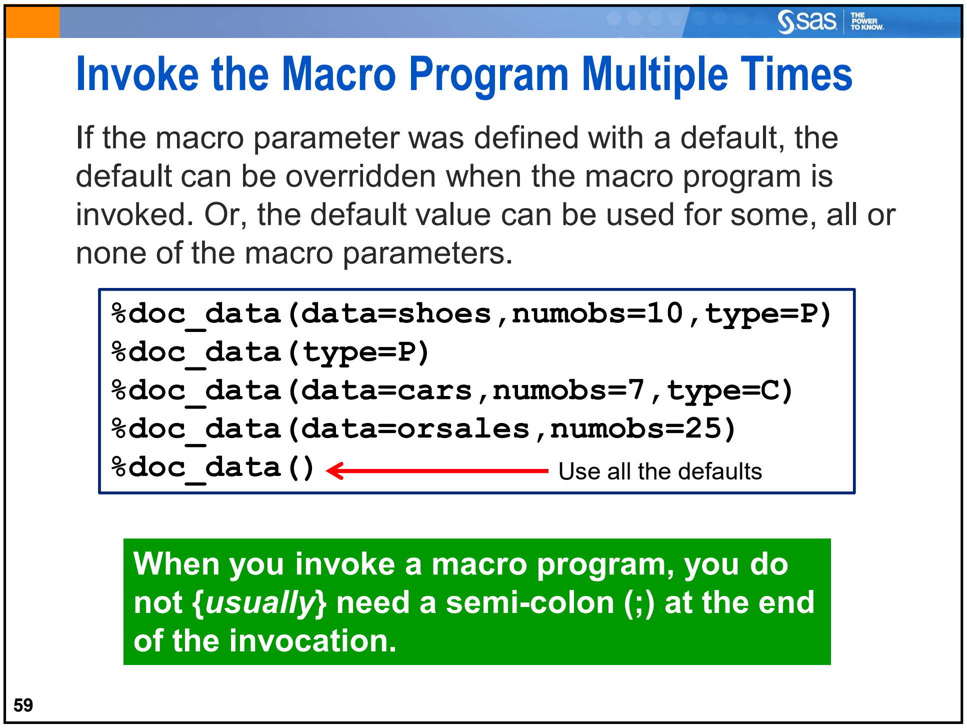
1) The macro processor will retrieve the %DOC_DATA macro program from the WORK.SASMACR catalog, and,

2) start typing code, based on any conditions, and

3) where there is a reference to a macro variable, the macro processor will do find-and-replace before it types the code.

4) When the macro processor is done, the typed code goes to the SAS compiler and then to be executed.

No & or % macro triggers remain in the code after the macro processor has finished code generation and resolution.



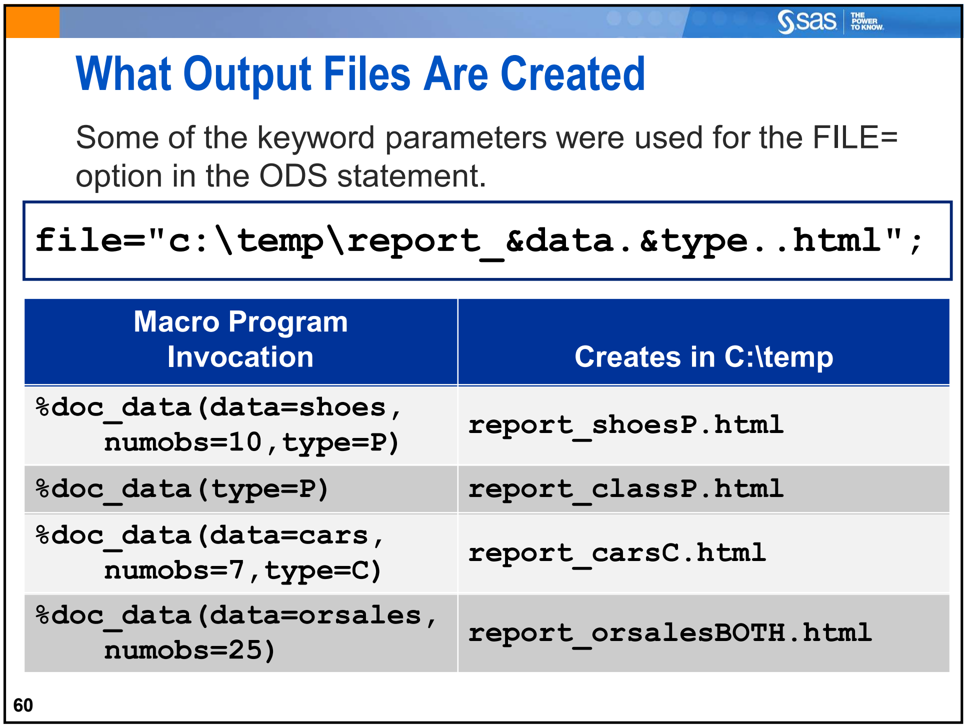
Invoke the Macro Program Multiple Times

If the macro parameter was defined with a default, the default can be overridden when the macro program is invoked. Or, the default value can be used for some, all or none of the macro parameters.

```
%doc_data(data=shoes,numobs=10,type=P)
%doc_data(type=P)
%doc_data(data=cars,numobs=7,type=C)
%doc_data(data=orsales,numobs=25)
%doc_data()
```

Use all the defaults

When you invoke a macro program, you do not {*usually*} need a semi-colon (;) at the end of the invocation.



What Output Files Are Created

Some of the keyword parameters were used for the FILE= option in the ODS statement.

```
file="c:\temp\report_&data.&type..html";
```

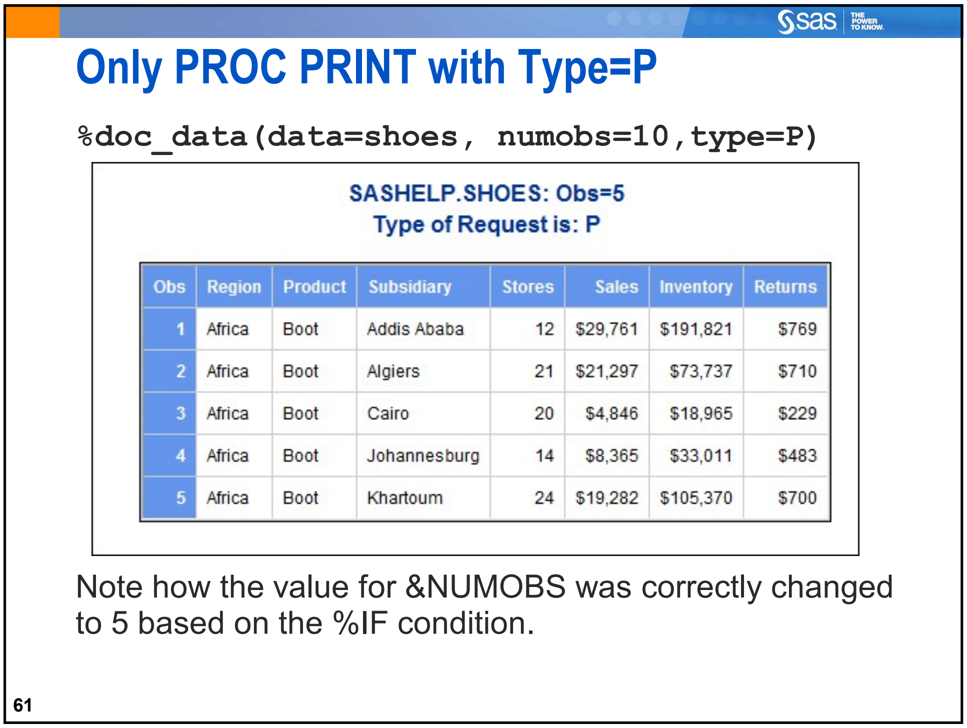
Macro Program Invocation	Creates in C:\temp

`%doc_data(data=shoes,`
`numobs=10,type=P)`
	`report_shoesP.html`

`%doc_data(type=P)`	`report_classP.html`

`%doc_data(data=cars,`
`numobs=7,type=C)`
	`report_carsC.html`

`%doc_data(data=orsales,`
`numobs=25)`
	`report_orsalesBOTH.html`



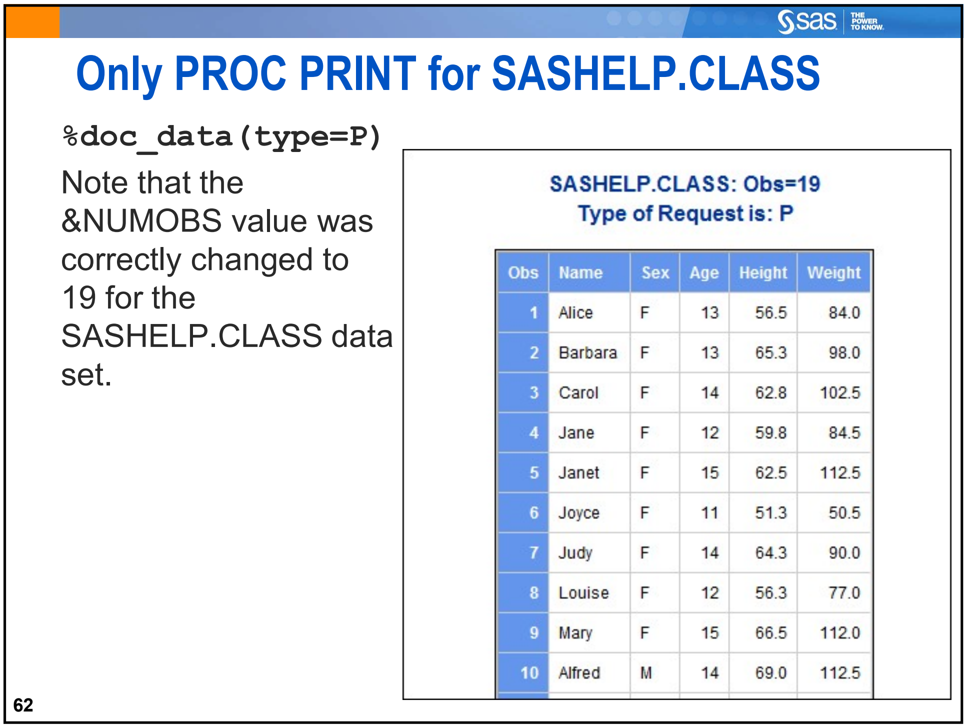
Only PROC PRINT with Type=P

```
%doc_data(data=shoes, numobs=10,type=P)
```

SASHELP.SHOES: Obs=5
Type of Request is: P

Obs	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	$29,761	$191,821	$769
2	Africa	Boot	Algiers	21	$21,297	$73,737	$710
3	Africa	Boot	Cairo	20	$4,846	$18,965	$229
4	Africa	Boot	Johannesburg	14	$8,365	$33,011	$483
5	Africa	Boot	Khartoum	24	$19,282	$105,370	$700

Note how the value for &NUMOBS was correctly changed to 5 based on the %IF condition.



Only PROC PRINT for SASHELP.CLASS

```
%doc_data(type=P)
```

Note that the &NUMOBS value was correctly changed to 19 for the SASHELP.CLASS data set.

SASHELP.CLASS: Obs=19
Type of Request is: P

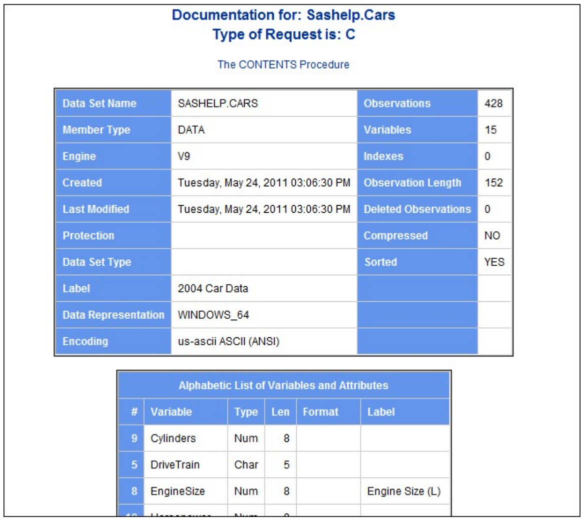
Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0
2	Barbara	F	13	65.3	98.0
3	Carol	F	14	62.8	102.5
4	Jane	F	12	59.8	84.5
5	Janet	F	15	62.5	112.5
6	Joyce	F	11	51.3	50.5
7	Judy	F	14	64.3	90.0
8	Louise	F	12	56.3	77.0
9	Mary	F	15	66.5	112.0
10	Alfred	M	14	69.0	112.5

PROC CONTENTS for SASHELP.CARS

```
%doc_data(data=cars,numobs=7,type=C)
```

Note the PROC CONTENTS was generated for the SASHELP.CARS data set.

(&NUMOBS was specified but not used.)



Documentation for: Sashelp.Cars
Type of Request is: C

The CONTENTS Procedure

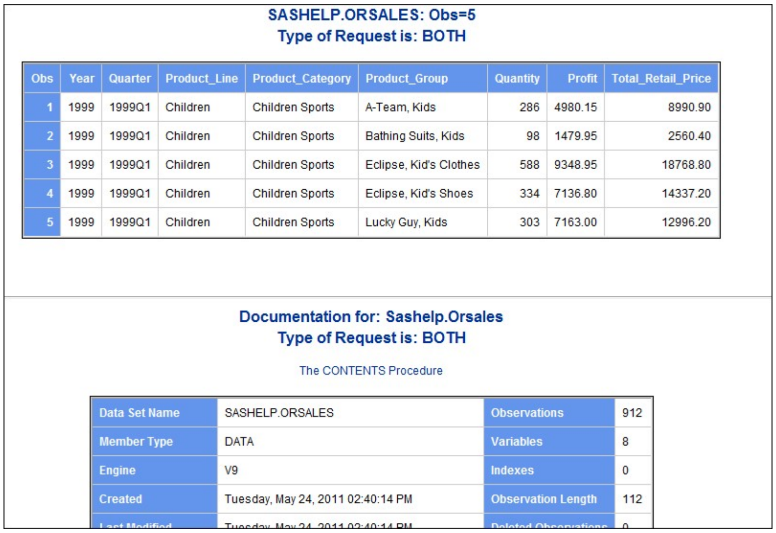
Data Set Name	SASHELP.CARS	Observations	428
Member Type	DATA	Variables	15
Engine	V9	Indexes	0
Created	Tuesday, May 24, 2011 03:06:30 PM	Observation Length	152
Last Modified	Tuesday, May 24, 2011 03:06:30 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label	2004 Car Data		
Data Representation	WINDOWS_64		
Encoding	us-ascii ASCII (ANSI)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
9	Cylinders	Num	8		
5	DriveTrain	Char	5		
8	EngineSize	Num	8		Engine Size (L)

Both Reports for SASHELP.ORSALES

```
%doc_data(data=orsales, numobs=25)
```



SASHELP.ORSALES: Obs=5
Type of Request is: BOTH

Obs	Year	Quarter	Product_Line	Product_Category	Product_Group	Quantity	Profit	Total_Retail_Price
1	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90
2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40
3	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Clothes	588	9348.95	18768.80
4	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Shoes	334	7136.80	14337.20
5	1999	1999Q1	Children	Children Sports	Lucky Guy, Kids	303	7163.00	12996.20

Documentation for: Sashelp.Orsales
Type of Request is: BOTH

The CONTENTS Procedure

Data Set Name	SASHELP.ORSALES	Observations	912
Member Type	DATA	Variables	8
Engine	V9	Indexes	0
Created	Tuesday, May 24, 2011 02:40:14 PM	Observation Length	112

Debugging Tips

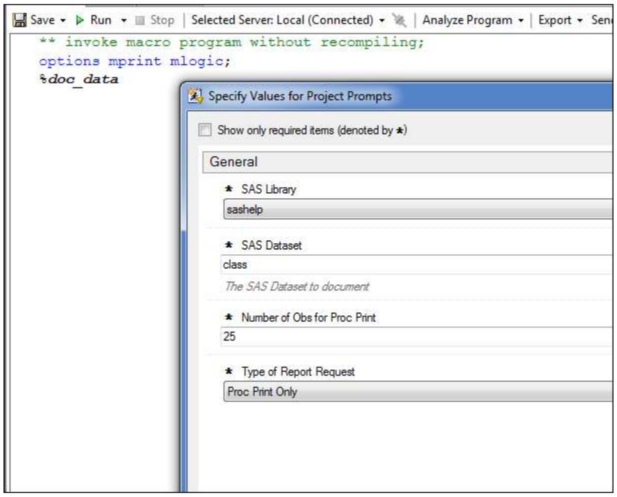
»To avoid debugging frustration, always start with a **working** SAS program!

Use the macro diagnostic options:

```
options mcompilenote=all mprint mlogic symbolgen;
%doc_data(data=orsales,numobs=25)
```

Write Once, Run Many Times

Once the macro program definition has been written and stored in a macro catalog, you can call it many times (especially if you store it in an "autocall" macro catalog).


Save
Run
Stop
Selected Server: Local (Connected)
Analyze Program
Export
Sen
Specify Values for Project Prompts
General

```
** invoke macro program without recompiling;
options mprint mlogic;
%doc_data
```

☐ Show only required items (denoted by *)

* SAS Library
sashelp

* SAS Dataset
class
The SAS Dataset to document

* Number of Obs for Proc Print
25

* Type of Report Request
Proc Print Only

Conclusion

The truth is that there's a lot more to learn about the macro facility and even cooler stuff to learn.

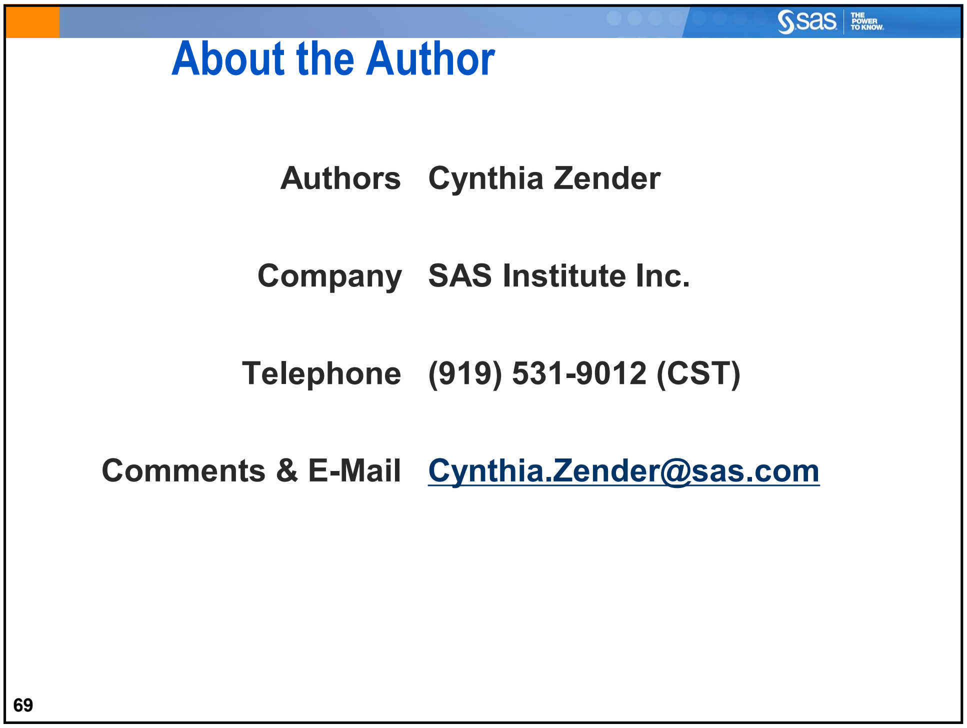
Start slowly, build your confidence, take our excellent SAS macro classes, read some user group papers and experiment.

Remember, "what happens in macro, stays in macro" – the SAS compiler never sees those **&** or **%** symbols unless you want it to!

Your Turn

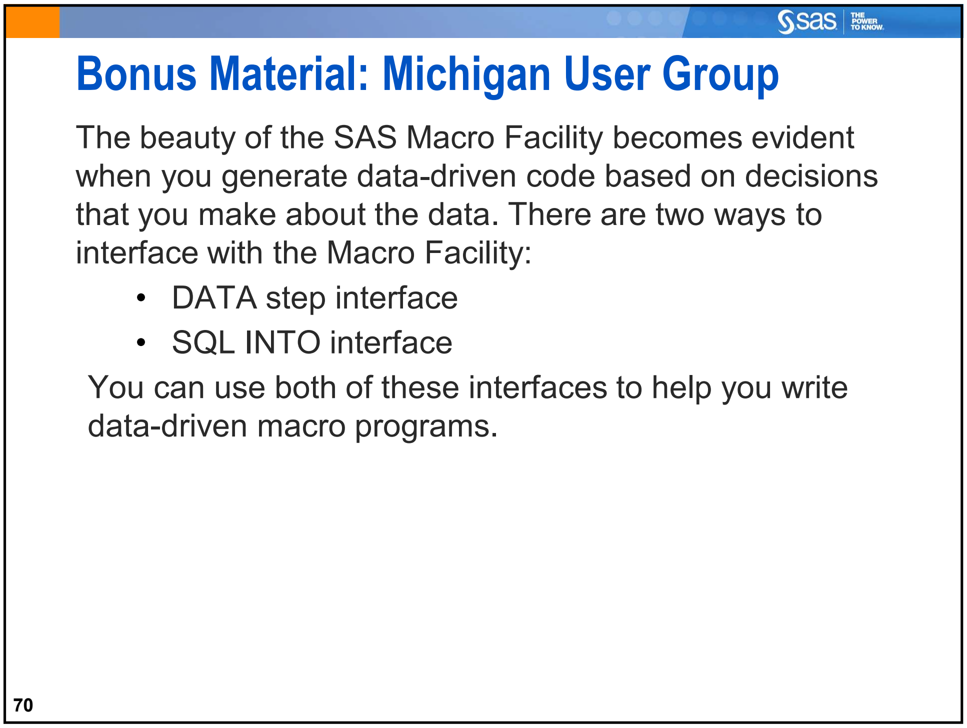
Questions

?



About the Author

Authors	**Cynthia Zender**
Company	**SAS Institute Inc.**
Telephone	**(919) 531-9012 (CST)**
Comments & E-Mail	**Cynthia.Zender@sas.com**

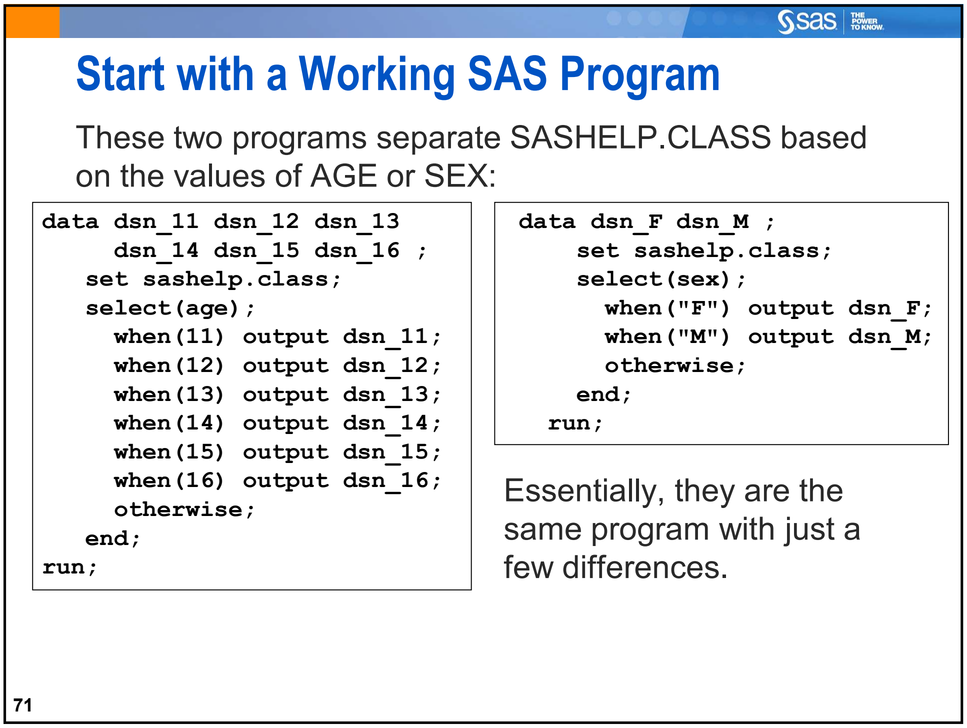


Bonus Material: Michigan User Group

The beauty of the SAS Macro Facility becomes evident when you generate data-driven code based on decisions that you make about the data. There are two ways to interface with the Macro Facility:

- DATA step interface
- SQL INTO interface

You can use both of these interfaces to help you write data-driven macro programs.



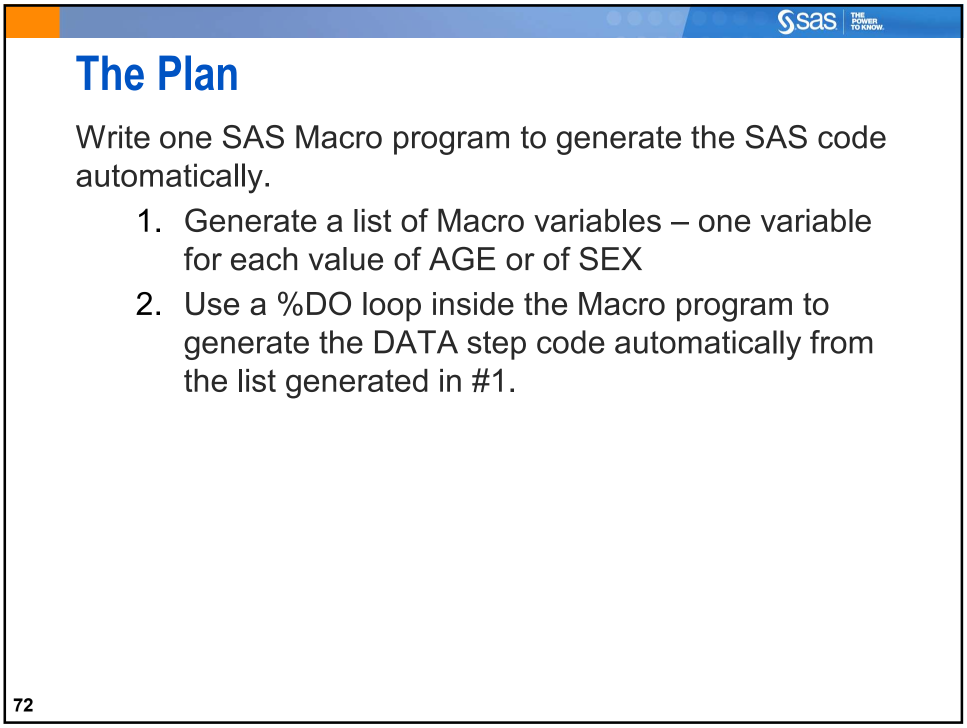
Start with a Working SAS Program

These two programs separate SASHELP.CLASS based on the values of AGE or SEX:

```
data dsn_11 dsn_12 dsn_13
     dsn_14 dsn_15 dsn_16 ;
   set sashelp.class;
   select(age);
     when(11) output dsn_11;
     when(12) output dsn_12;
     when(13) output dsn_13;
     when(14) output dsn_14;
     when(15) output dsn_15;
     when(16) output dsn_16;
     otherwise;
   end;
run;
```

```
data dsn_F dsn_M ;
    set sashelp.class;
    select(sex);
      when("F") output dsn_F;
      when("M") output dsn_M;
      otherwise;
    end;
  run;
```

Essentially, they are the same program with just a few differences.



The Plan

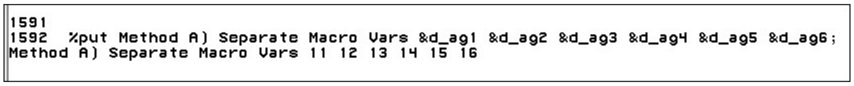
Write one SAS Macro program to generate the SAS code automatically.

1. Generate a list of Macro variables – one variable for each value of AGE or of SEX
2. Use a %DO loop inside the Macro program to generate the DATA step code automatically from the list generated in #1.

Compare Both Methods

DATA Step Method

```
proc sort data=sashelp.class(keep=age) out=values nodupkey;
     by age;
run;

data _null_;
   set values end=last;
   call symputx('d_ag'||left(put(_n_,2.0)), put(age,2.0));
   if last then do;
     call symputx('count',_n_);
   end;
run;

%put Method A) Separate Macro Vars &d_ag1 &d_ag2 &d_ag3
 &d_ag4 &d_ag5 &d_ag6;
```

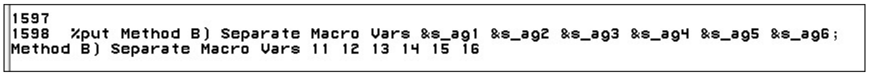


```
1591
1592  %put Method A) Separate Macro Vars &d_ag1 &d_ag2 &d_ag3 &d_ag4 &d_ag5 &d_ag6;
Method A) Separate Macro Vars 11 12 13 14 15 16
```

Compare Both Methods

SQL Method

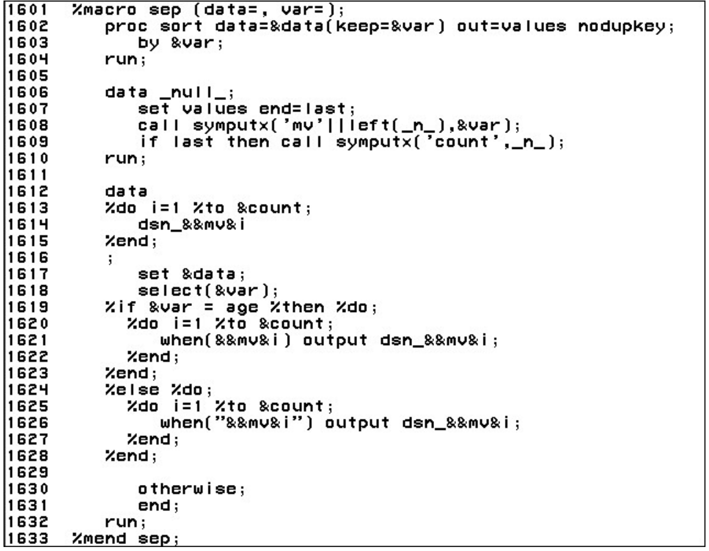
```
proc sql noprint ;
   select distinct(age) into :s_ag1-:s_ag6
      from sashelp.class;
quit;

%put Method B) Separate Macro Vars &s_ag1 &s_ag2 &s_ag3
&s_ag4 &s_ag5 &s_ag6;
```



```
1597
1598  %put Method B) Separate Macro Vars &s_ag1 &s_ag2 &s_ag3 &s_ag4 &s_ag5 &s_ag6;
Method B) Separate Macro Vars 11 12 13 14 15 16
```

Macro Program Definition

Here’s the full macro program definition:



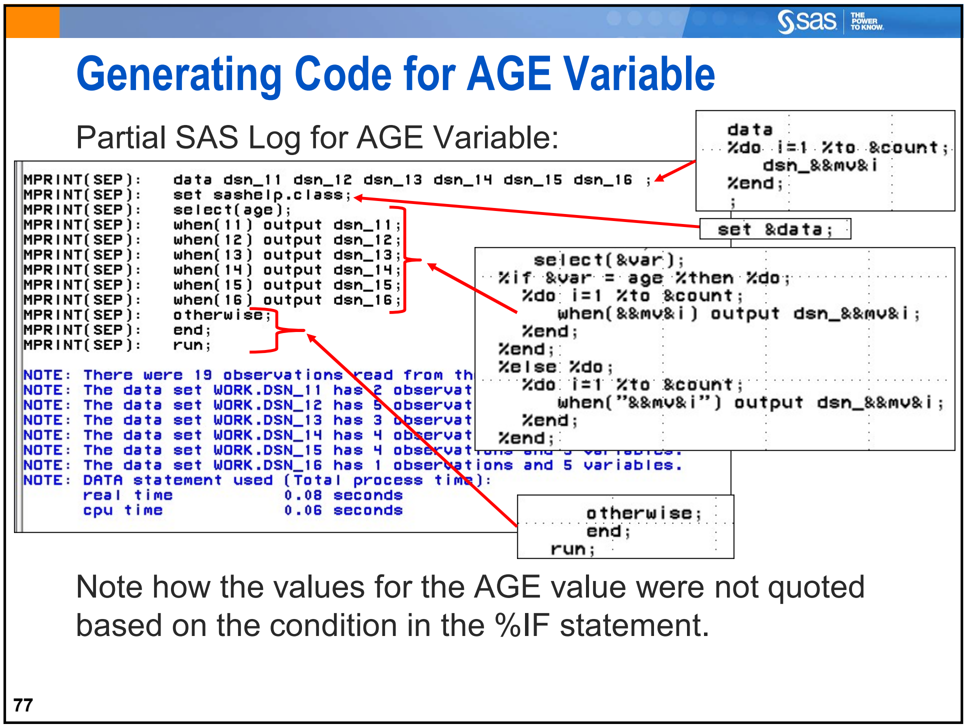
```
%macro sep (data=, var=);
   proc sort data=&data(keep=&var) out=values nodupkey;
      by &var;
   run;

   data _null_;
      set values end=last;
      call symputx('mv'||left(_n_),&var);
      if last then call symputx('count',_n_);
   run;

   data
   %do i=1 %to &count;
      dsn_&&mv&i
   %end;
   ;
      set &data;
      select(&var);
   %if &var = age %then %do;
     %do i=1 %to &count;
        when(&&mv&i) output dsn_&&mv&i;
     %end;
   %end;
   %else %do;
     %do i=1 %to &count;
        when("&&mv&i") output dsn_&&mv&i;
     %end;
   %end;

      otherwise;
      end;
   run;
%mend sep;
```

Invoking the Macro Program

To invoke the macro program, we use the macro name preceded with a %:

```
%sep(param=value, param=value)
```

```
options mprint;

%sep(data=sashelp.class, var=age)
%sep(data=sashelp.class, var=sex)
```



Generating Code for AGE Variable

Partial SAS Log for AGE Variable:

```
MPRINT(SEP):   data dsn_11 dsn_12 dsn_13 dsn_14 dsn_15 dsn_16 ;
MPRINT(SEP):   set sashelp.class;
MPRINT(SEP):   select(age);
MPRINT(SEP):   when(11) output dsn_11;
MPRINT(SEP):   when(12) output dsn_12;
MPRINT(SEP):   when(13) output dsn_13;
MPRINT(SEP):   when(14) output dsn_14;
MPRINT(SEP):   when(15) output dsn_15;
MPRINT(SEP):   when(16) output dsn_16;
MPRINT(SEP):   otherwise;
MPRINT(SEP):   end;
MPRINT(SEP):   run;

NOTE: There were 19 observations read from th
NOTE: The data set WORK.DSN_11 has 2 observat
NOTE: The data set WORK.DSN_12 has 5 observat
NOTE: The data set WORK.DSN_13 has 3 observat
NOTE: The data set WORK.DSN_14 has 4 observat
NOTE: The data set WORK.DSN_15 has 4 observat
NOTE: The data set WORK.DSN_16 has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.08 seconds
      cpu time            0.06 seconds
```

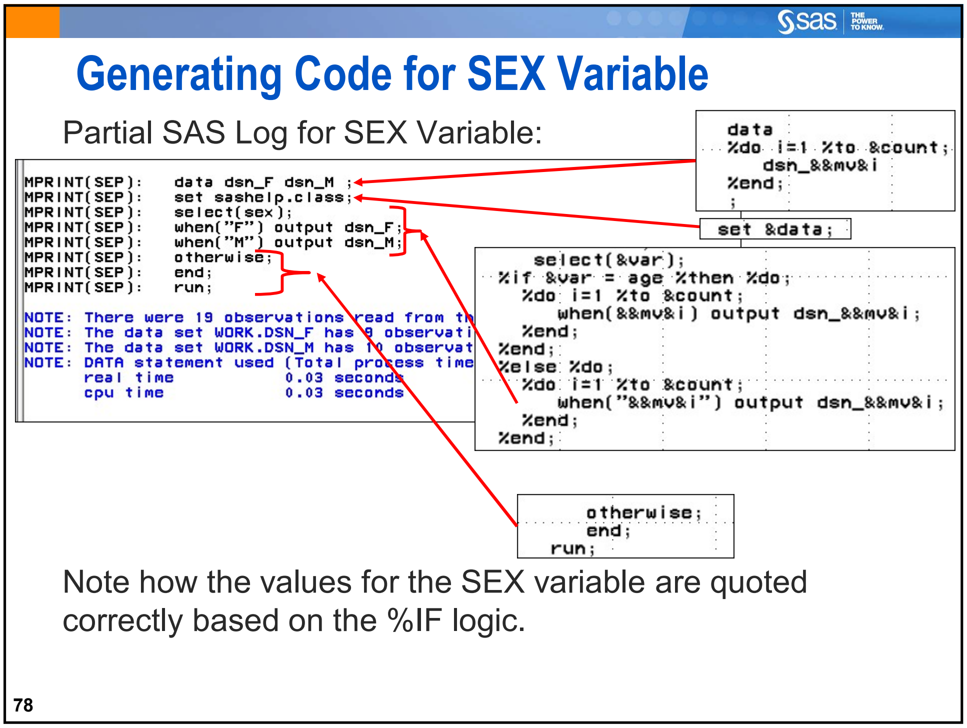
```
data
%do i=1 %to &count;
   dsn_&&mv&i
%end;
;
```

```
set &data;
```

```
    select(&var);
%if &var = age %then %do;
  %do i=1 %to &count;
     when(&&mv&i) output dsn_&&mv&i;
  %end;
%end;
%else %do;
  %do i=1 %to &count;
     when("&&mv&i") output dsn_&&mv&i;
  %end;
%end;
```

```
      otherwise;
      end;
   run;
```

Note how the values for the AGE value were not quoted based on the condition in the %IF statement.



Generating Code for SEX Variable

Partial SAS Log for SEX Variable:

```
MPRINT(SEP):   data dsn_F dsn_M ;
MPRINT(SEP):   set sashelp.class;
MPRINT(SEP):   select(sex);
MPRINT(SEP):   when("F") output dsn_F;
MPRINT(SEP):   when("M") output dsn_M;
MPRINT(SEP):   otherwise;
MPRINT(SEP):   end;
MPRINT(SEP):   run;

NOTE: There were 19 observations read from th
NOTE: The data set WORK.DSN_F has 9 observati
NOTE: The data set WORK.DSN_M has 10 observat
NOTE: DATA statement used (Total process time
      real time           0.03 seconds
      cpu time            0.03 seconds
```

```
data
%do i=1 %to &count;
   dsn_&&mv&i
%end;
;
```

```
set &data;
```

```
    select(&var);
%if &var = age %then %do;
  %do i=1 %to &count;
     when(&&mv&i) output dsn_&&mv&i;
  %end;
%end;
%else %do;
  %do i=1 %to &count;
     when("&&mv&i") output dsn_&&mv&i;
  %end;
%end;
```

```
      otherwise;
      end;
   run;
```

Note how the values for the SEX variable are quoted correctly based on the %IF logic.

Final Thoughts

The SAS Macro Facility is extremely powerful. But it only types code for you! Therefore, it is essential to start with a working SAS program or programs before you write any macro code.

Classes:
SAS Macro Language 1: Essentials
SAS Macro Language 2: Advanced Techniques

Books:
Art Carpenter
Michele Burlew

Papers:
http://www.lexjansen.com (to search for papers on Macro topics)